Umgang und Optimierungsmöglichkeiten datenintensiver Prozesse in Serviceorientierten Architekturen (SOA)

Bachelorarbeit

Zur Erlangung des Akademischen
Grades Bachelor of Engineering "B. Eng."

Technische Fachhochschule Wildau

Fachbereich Ingenieurwesen
Studiengang Telematik

vorgelegt von:	**Timo Jonathan Schmidt**
Reg.-Nr.:	**T06/16/SS2009**
Betreuer:	**Prof. Dr. rer. nat. Janett Mohnke,** **Dipl.-Mathematiker Bernd Weissbach**
Themenstellender Betrieb:	**DLR e.V.**
Betrieblicher Betreuer:	**M. Eng. Sten Ruppe**

Bibliographische Beschreibung und Referat

Umgang und Optimierungsmöglichkeiten datenintensiver Prozesse in Serviceorientierten Architekturen (SOA) Bachelorarbeit, Technische Fachhochschule Wildau 2009, 76 Seiten, 21 Abbildungen, 43 Literaturangaben, 9 Anlagen, 1 Beilage

Ziel

Ziel der Arbeit ist der adäquate Umgang und die optimierte Verwendung datenintensiver Prozesse sowie die Verminderung der Datenmenge bei Übertragungen und die verbesserte Datenorganisation in einer Serviceorientierten Architektur.

Inhalt

Bei dieser Arbeit werden die Eigenschaften und Funktionalitäten einer SOA vorgestellt und Voraussetzungen für eine richtige Verwendung genannt.
Es werden verbesserte Datenhaltung und -management durch die Partitionierung der Datenbanktabellen von Datenbankmanagementsystemen behandelt.
Für die Optimierung von Übertragungsmengen wird JSON als geeignetes Übertragungsformat alternativ zu XML, GZIP zum Komprimieren von Datenmengen, sowie der richtige Umgang dargelegt.
Eine einfache und automatische Erzeugung von Web Services wird anhand Apache Ant und Apache Axis2 bearbeitet.

Selbständigkeitserklärung

Hiermit versichere ich, die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Wildau, den 12. August 2009

Timo Jonathan Schmidt

Abkürzungsverzeichnis

AAR Axis Archive
AXIS Apache eXtensible Interaction System
BPEL Business Process Execution Language
BPEL4WS Business Process Execution Language for Web Services
DLR Deutsches Zentrum für Luft- und Raumfahrt
DTD Document Type Description
EAI Enterprise Application Integration
ESB Enterprise Service Bus
FCD Floating Car Data
HTML Hypertext Markup Language
HTTP Hypertext Transfer Protocol
IP Internet Protocol
IT Informationstechnik
JAR Java Archive
OM Object Model
PDF Portable Document Forward
POJO Plain Old Java Object
SAX Simple API for XML
SOA Serviceorientierte Architektur
SOAP Simple Object Access Protocol
SQL Structured Query Language
StAX Streaming API for XML
TCP Transmission Control Protocol
TDP Traffic-Data-Platform
UDDI Universal Description, Discovery and Integration
URI Uniform Resource Identifier
URL Uniform Resource Locator
W3C World Wide Web Consortium
WAR Web Archive

WSDL Web Service Description Language
WS-BPEL Web Service Business Process Execution Language
WS-CDL Web Service Choreography Description Language
WSCI Web Service Choreography Interface
XML Extensible Markup Language
XSD XML-Schema-Definition
XSL Extensible Stylesheet Language
XSLT XSL-Transformation

Inhaltsverzeichnis

1. Einleitung

1.1. Gegenstand und Motivation

In der Systemstruktur eines Unternehmens steigt die Größe und die Heterogenität im Laufe der Zeit kontinuierlich an, neue Systeme, Komponenten mit unterschiedlichen Eigenheiten, wie zum Beispiel Programmiersprachen, Modellierungssysteme und Softwarehersteller kommen hinzu. Daraus resultiert eine Zunahme der Komplexität, Inflexibilität und Schwere der Integration für neue Komponenten. Es werden strategische Vorgehensweisen und Denkverhalten insistiert, die speziell die Agilität und Flexibilität fördern und letztendlich dem Unternehmen eine Reduzierung der Kosten für Wartung und Entwicklung ermöglichen.
Weiterhin wird eine hohe Interoperabilität verteilter Systeme gefordert, in denen Daten räumlich verteilt liegen und auch in Bezug auf ihre Datenquellen sich inhaltlich differenzieren können. Funktionalitäten sollen untereinander agieren und von jedem System aus verfügbar und nutzbar sein, dadurch ist ein automatischer Abgleich der Daten möglich. Unternehmensübergreifende Interaktion und Integration von Geschäftsprozessen nimmt aus Unternehmersicht signifikante Bedeutung an. Aus technischer Sichtweise ist ein Prozess der Verlauf und die Abfolge von Funktionen und Aktivitäten. Für die unternehmensübergreifende Interaktion von Prozessen sind mittlerweile Web Services eine etabilierte Technologie, die Dank der zugrunde liegenden Standards Interoperabilitätsprobleme lösen. Doch zuerst müssen Prozesse als Services abgebildet werden können.
Die Serviceorientierte Architektur (SOA) stellt in diesem Zusammenhang ein bedeutendes Systemparadigma dar. Prozesse und ihre Funktionen werden mit hoher Abstraktion zu losen gekoppelten Services aufgeteilt und definiert. Eine lose Kopplung meint, dass Services nicht fest an einer Aufgabe gebunden werden. Durch die Aufteilung und Abstraktion wird die Wiederverwendbarkeit unterstützt und Redundanzen werden vermieden. Die lose Kopplung ermöglicht ein dynamisches Ersetzen, Erneuern oder Binden von Services je nach Anforderung an das System. Ein weiterer Vorteil ist gezielt und besser skalieren zu können, indem Services je nach Aufwand unterschiedliche Prozessoren zur Abarbeitung zugeordnet werden [Vgl.: Reitbauer und Novakovic 2009]. SOA unterliegen in Unternehmen großer Nachfrage und die Investitionen in SOA-Landschaften nehmen zu. Die Ausgaben von Unternehmen für SOA-Komponenten und -Leistungen betrugen im Jahr 2007

zwei Milliarden Dollar und sollen bis 2014 auf 9,1 Milliarden Dollar ansteigen [WinterGreen April 2008].
Laut einer aktuellen Studie, dem SOA Check 2009, wird SOA derzeit von 47 % der 111 befragten Personen aus großen bis mittelständischen Unternehmen in Deutschland, Schweiz und Österreich eingesetzt, und von 37 % wird der Einsatz geplant. Nur 16 % beschäftigen sich nicht mit dem Einsatz [It-Research u. a. 2009].
Eine Identifizierung von Services aus Geschäftsprozessen wird als Geschäftsprozessmanagement bezeichnet. Der Zusammenhang zwischen SOA und Geschäftsprozessmanagement ist somit prägnant. Um Prozesse hinsichtlich ihrer Abläufe und Aktivitäten zu kontrollieren und zu erfassen, ist ein Geschäftsprozessmanagement unabdingbar. Durch Geschäftsprozessmodellierung wird die Orchestrierung von Prozessen, also die Komposition möglich. Eine wichtige Technologie, die dem Geschäftsprozessmanagement als Werkzeug dient, ist BPEL (Business Process Execution Language), eine auf XML-basierte Sprache, die sich durch das vereinfachte Erfassen und Definieren von Prozessen und die Ablaufsteuerung und Durchführung auszeichnet.
Unternehmen ist der richtige Umgang mit datenintesiven Prozessen von hoher Wichtigkeit, denn diese stellen für eine Unternehmung neue Herausforderungen bei der Datenorganisation und -übertragung dar. Im Bezug auf unternehmensübergreifende Interaktionen mit Web Services zeigt sich vor allem die Reduzierung von Übertragungsinhalt bei datenintensiven Prozessen als substanziell. Bestandteil dieses Aspektes ist die optimale Erzeugung eines Web Services und die richtige Nutzung der zugrunde liegenden Standards.

Hieraus entstand im Institut für Verkehrssystemtechnik des DLR e.V. die Motivation zu einer Serviceorientierten Architektur in einer neuen Plattform, die Traffic-Data-Platform (TDP). Unabhängig von verteilten Systemen und Daten soll in der Plattform der Zugriff und Austausch von Verkehrsdaten erleichtert werden. Die Motivationspunkte und Vorteile der TDP werden folgend genannt:

- **Unterstützung agiler Software Entwicklung**
- **Dynamisches System**
- **Förderung der Interoperabilität externer/interner Funktionalitäten**
- **Erleichterte Erweiterbarkeit, Erneuerung, Wartung**
- **Bessere Integrations-, Kompatibilitäts-, und Funktionstests**
  Durch die Abstraktion von Funktionalitäten in Services können Tests gut vor wirklicher Integration in das System verwirklicht werden. Benötigte Funktionen sind schon vorab verfügbar und verwendbar für Testdurchführungen.

Bei den Verkehrsdaten handelt es sich beispielsweise um Floating Car Data (FCD) und Schleifendaten aus vielen verschiedenen Bereichen, die eine beträchtliche Datenmenge produzieren, welche übertragen, organisiert und gespeichert werden muss. Daher nimmt die Optimierung datenintensiver Prozesse in der Plattform eine bedeutende Aufgabe ein.

1.2. Anforderungen und Zielsetzung

Eine mittlerweile etablierte Technologie verteilter Anwendungen sind Web Services, die Dank der zugrunde liegenden Standards Interoperabilitätsprobleme lösen. Für den Zugriff und Austausch von Verkehrsdaten (z.B. Schleifendaten, FCD, Kameradaten, prozessierte und fusionierte Traffic-Daten, etc.) hat das Institut für Verkehrssystemtechnik des DLR e.V. eine Traffic-Data-Platform (TDP) im Entwicklungsstadium, welche auf die Technologie der Web Services aufsetzt. Mit der zunehmenden Datenmenge und deren Integration in die Plattform entstehen neue Herausforderungen hinsichtlich der Datenorganisation, -speicherung und -übertragung.
Im Rahmen dieser Bachelorarbeit werden die TDP und deren Serviceorientierte Architektur für die Verwendung optimiert und für die Datenorganisation Verbesserungen erarbeitet. Für eine Optimierung von datenintensiven Prozessen ist eine Reduzierung der Datenmengen vorrangig zu betrachten. Im Zusammenhang mit Web Service Technologien werden für die Übertragung großer Datenmengen Konzepte erläutert und bewertet. Die Bewertung wird anhand der fähigen Integration in die TDP und des allgemeinen Nutzenvorteils vollzogen. Hinsichtlich der Datenorganisation wird ein Konzept ausgearbeitet, das für die Speicherung und Verwendung von hohen Datenaufkommen vorteilhaft angewendet werden kann. Das Konzept wird prototypisch in Bezug auf die TDP und die fähige Integration durchgeführt. Für zukünftige Betrachtungen werden Recherchen bezüglich des Managements von Prozessen durchgeführt. Durch diese Recherchen können Informationen bereitgestellt werden, um eine Komposition und Kontrollen und somit das Steuern von Prozessen in der TDP zu ermöglichen.

1.3. Gliederung der Arbeit

Der von der Systemanalyse bis hin zur Systementwicklung und -weiterentwicklung ausgearbeitete Plan erfolgt für diese Arbeit in Form des V-Modells. Dieses Vorgehensmodell gibt eine standardisierte Vorgehensweise und Rollenzuweisung vor. Dadurch wird die Erfolgswahrscheinlichkeit für die letztendliche Abnahme und Nutzung gesteigert. Das Modell beschreibt den Entwicklungsablauf in Abhängigkeit von Zeit und Detaillierung. In der Abbildung 1.1 wird das V-Modell dargestellt.
Die detaillierte Beschreibung des Modells und die Bewertung dieser Vorgehensweise

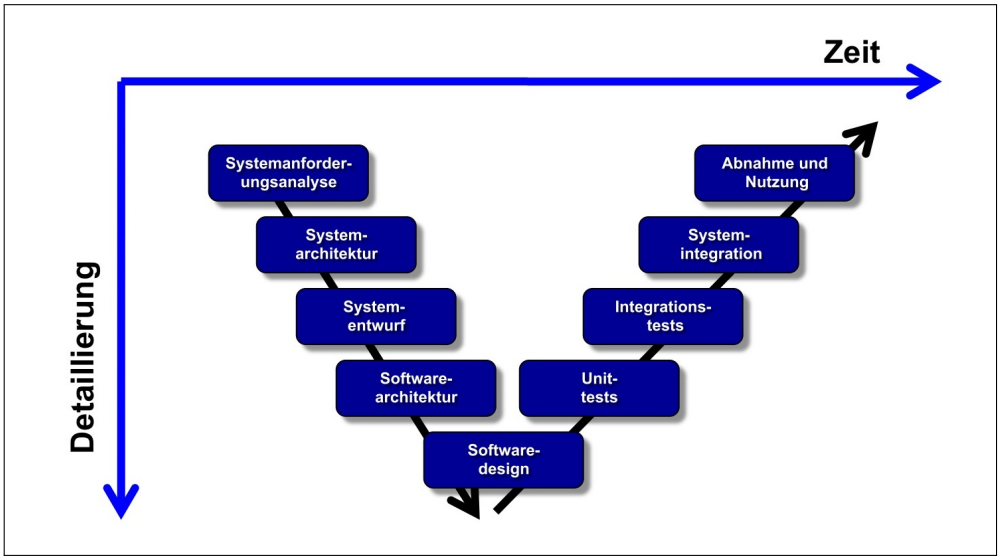


Abbildung 1.1.: Projektplan und -durchführung nach V-Modell
Quelle: [PMH 2008]

erfolgt in Kapitel 7.1. Die Arbeit ist thematisch in die Abschnitte Grundlagen, Konzepte (Durchführung) und Bewertung aufgeteilt. Die Grundlagen in Kapitel 2 umfassen die relevanten Informationen und Recherchen für diese Arbeit und beinhalten somit die Basis für die Analyse, Konzepte und Durchführung in den Kapiteln 3, 4, 5 und 6. In Kapitel 3 wird detailliert auf die TDP eingegangen. Anhand dieser Beschreibung und den Zielen werden in den Kapiteln 4, 5 und 6 Lösungsansätze entworfen und dokumentiert. In Kapitel 7 werden die allgemeine Vorgehensweise und die daraus resultierenden Entwicklungen und Konzepte bewertet.

1.4. Abgrenzung der Arbeit

Im Vergleich zu anderen Ausarbeitungen der Thematik SOA wird in dieser Arbeit auf Basis der Grundlagen vorwiegend der Aspekt von Optimierungsmöglichkeiten im Datenumgang und -speicherung von datenintensiven Prozessen behandelt. Auf Grund dessen werden verbesserte Übertragungsmechanismen und Datenorganisationsysteme als Lösungsansätze konzipiert. Die datenintensiven Prozesse werden primär durch eine Reduzierung der Daten verbessert. Sekundär werden sie durch eine Komposititon (Orchestrierung), Ablaufsteuerung und Management von Prozessen optimiert. Infolgedessen werden für diese Arbeit Konzepte und Implementierungen für eine geeignete Verminderung der Datenmengen erarbeitet, nicht aber für eine Dienstkomposition. Es werden ausgiebige Recherchen für die sekundären

Betrachtungen[1] durchgeführt und dargelegt. Des Weiteren können anhand dieser Informationen die zukünftigen Ausblicke und Empfehlungen aufgestellt werden.
Diese Ausarbeitungen erfolgen für das Institut für Verkehrssystemtechnik des DLR e.V. und deren neuartige Plattform im Entwicklungsstadium, die TDP. In dieser Arbeit werden folgende Zusammenhänge und Thematiken erwähnt, aber nicht im Einzelnen behandelt: Die Zusammenhänge und Details von Hypertext Makeup Language (HTML) und Web Service Technologien werden nicht erarbeitet. Für die Web Services Datenübertragung und SOA Optimierung ist es irrelevant. Es wird nicht auf die Veränderung, Hinzufügung oder Löschung von Partitionen eingegangen (Kapitel 2.4, Datenorganisation und -speicherung). Partitionierung wird nur als Entwurf und Lösungsansatz für bessere Datenorganisation vorgestellt. In dem Zusammenhang werden in dieser Arbeit Funktionen, Vorteile und Nutzen für eine Aufteilung von Datenbanktabellen erläutert. Es ist daher nicht notwendig verwaltungstechnische Aspekte zu berücksichtigen. Von den vier Methoden der Partitionierung wird in Kapitel 4.3 nur die Funktion *RANGE* anhand eines Beispiels in Bezug auf MySQL veranschaulicht. Zudem wird ein weiteres Beispiel mit einer *RANGE*-Partitionierung gezeigt, bei dem eine Unterpartitionierung mit der *HASH* Funktion vorliegt. Es genügt eine Funktionalität bildlich darzustellen um den Aufbau und die Benutzung zu beschreiben. In Kapitel 3.3 wird auf die Verwendung von Axis2 und Ant in der TDP eingegangen und in Kapitel 6.1 eine Anwendung und Entwicklung mit diesen dokumentiert. Teil der Arbeit ist nicht die Beschreibung einer Installation dieser Frameworks von Apache. Sollte an dieser Stelle dennoch Interesse dafür bestehen, wird empfohlen entsprechende Informationen auf den Internetseiten des Anbieters Apache zu verwenden[2].
Die Aufgabenstellung fordert ein Konzept zur Übertragung großer Datenmengen auf Basis bestehender Web Service-Technologien zu recherchieren. An dieser Stelle werden die Beispiele WSBPEL 2.0, Dienstkomposition durch Orchestrierung oder Choreographie genannt. Anhand der Recherchen ergibt sich, dass diese Technologien für die Optimierung der Übertragung großer Datenmengen nicht ausschlaggebend sind. Sie nehmen stattdessen für die Ablaufsteuerung, Kontrolle und Management von Prozessen eine bedeutende Rolle ein. Auf Grund dessen werden die Technologien und das Management von Prozessen in dieser Arbeit erläutert. Für zukünftige Betrachtungen und Optimierung nehmen diese einen hohen Stellenwert ein.

[1] wie zum Beispiel Geschäftsprozessmanagement, Dienstkomposition durch Orchestrierung (mit WS-BPEL) und Choreographie (mit WS-CDL)
[2] siehe: [Apache 2009b] und [Apache 2009a]

2. Fundamentale Grundlagen

2.1. Elementarwissen zu Serviceorientierten Architekturen (SOA)

In der Literatur [Nissen u. a. 29. Mai 2008, S. 4] wird SOA folglich definiert:

> *"Sammelbegriff für ein bestimmtes Strukturparadigma für (betriebliche) Anwendungssysteme, das gegenüber anderen Ansätzen der (Software-) Wiederverwendung auf den Aspekt der (Dienst-) Verwendung fokussiert."*

Eine Serviceorientierte Architektur ist ein Systemparadigma und -stil, nach dem Anwendungen, Schnittstellen und Funktionen abstrakt aufgeteilt und zu Services implementiert werden. Es wird keine Technologie vorgegeben, mit der das Konzept einer SOA umgesetzt werden muss. Dadurch wird die Heterogenität verteilter Systeme unterstützt. Den Leitgedanken bildet das dynamische Suchen, Binden und Nutzen von Services, die erst zur Laufzeit bekannt sein müssen. Denn diese Services liegen in der Architektur lose gekoppelt vor, wodurch eine dynamische Anpassung ermöglicht wird und die Flexibilität des Systems zunimmt.

2.1.1. Geschichte und Entwicklung

EAI (Enterprise Application Integration) kann als eine Art Vorgänger von SOA angesehen werden. Es ist ein Systemkonzept, mit dem in verteilten Systemen Applikationen und deren Prozesse und Daten zusammenhängend integriert werden.Im Vergleich zu SOA werden aber die Funktionen nicht als Services aufgeteilt, sondern Anwendungen werden, so wie sie vorliegen, gebunden.
Im April 1996 wurde SOA erstmalig von Gartner in einem Research beschrieben [Natis 16. April 2003]. Schon zu dieser Zeit erkannte man, was für ein erhebliches Potenzial und eine prägnante Bedeutung in diesem Konzept der Serviceorientierung liegen.

SOA aktuell

In Zeiten der Finanzkrise 2008, die aktuell im Jahr 2009 in eine weltweite Wirtschaftskrise ausartete, war anzunehmen, dass auch SOA-Vorhaben und -Investitionen davon betroffen sind und deutlich minimiert werden würden. Aber es zeigte sich, dass vor allem Firmen, die Serviceorientierte Architekturen in ihrer Unternehmensstruktur verwenden, in diesen Krisenzeiten wirtschaftliche Vorteile haben [Blogspan Magazine März 2009]. Denn diese Unternehmen verfügen durch SOA über eine höhere Flexibilität in ihren Systemen und damit über eine bessere und leichtere Kontrolle, Koordination und Auslagerung der Geschäftsprozesse. Diese Vorteile sind größtenteils auf das dynamische Konzept von SOA zurückzuführen. Infolgedessen werden trotz Wirtschaftskrise SOA-Vorhaben und -Investitionen ein noch stärkeres Wachstum prognostiziert. In Deutschland zeigen sich einige SOA-Initiativen, so wurde beispielsweise am 04. März 2008 die erste deutschsprachige und herstellerneutrale Informationsplattform[1] gegründet [Ziegler 04. März 2008]. Des Weiteren wird seit 2007 ein Status Quo von SOA durch It-Research, Wolfgang Martin Team und TU Darmstadt analysiert, was von den Institutionen T-Systems, Cordys, SAP und Informatica unterstützt wird. Dieser Research bewertet SOA für Deutschland, Österreich und Schweiz durch Umfragen von freiwilligen Teilnehmern[Vgl.: It-Research u. a. 2009].

2.1.2. Funktionsweise und Aufbau

Das Grundkonzept einer SOA besteht aus drei Einheiten, die bei Erstellung, Bereitstellung, Binden und Nutzen eines Services zusammen kooperieren und aus denen sich die Funktionsweise der Architektur ergibt. Allen voran steht ein Service Provider, der über die Ressource eines Dienstes verfügt und diese zur Verfügung stellt. Als Dienstanbieter ist er gleichzeitig für die Leistung und Qualität verantwortlich. Dabei ist es unerheblich, ob es sich beim Anbieter um den Entwickler handelt. Als Provider stellt er nicht nur den Dienst bereit, sondern veröffentlicht Informationen, die den Service auffindbar und erreichbar machen. Bei einer eingeschränkten Öffentlichkeit des Dienstes, zum Beispiel bei einem Angebot nur für registrierte Kunden, ist der Dienstanbieter für die Validierung einer Nutzer-Authentisierung zuständig. Der Service Broker ist eine Art Register, in dem verfügbare Services eingetragen werden können. Dabei wird nicht der Service, die eigentliche Dienstleistung registriert, sondern nur Informationen und Beschreibungen. Diese werden vom Provider veröffentlicht. Des Weiteren sind im Register Daten verzeichnet, die ein Ausfindigmachen und Benutzen eines Services erzielen. Es können auch optional Informationen, wie Ansprechpartner und Emailadressen, für eine Kontaktaufnahme des Nutzers zum Provider enthalten sein, sowie Informationen für eine Identifikation des Providers. Ein Service Consumer kommuniziert mit dem Service Broker um

[1] unter der Webadresse http://www.SOA-Know-How.de (Letzter Zugriff Juli 2009)

eine gewünschte Dienstleistung ausfindig zu machen. Der Consumer kann aus dem Broker spezifische Daten über einen Service erhalten, die dazu dienen eine Verbindung herzustellen. In Abbildung 2.1 ist das Zusammenspiel dieser drei elementaren Bestandteile einer SOA dargestellt.

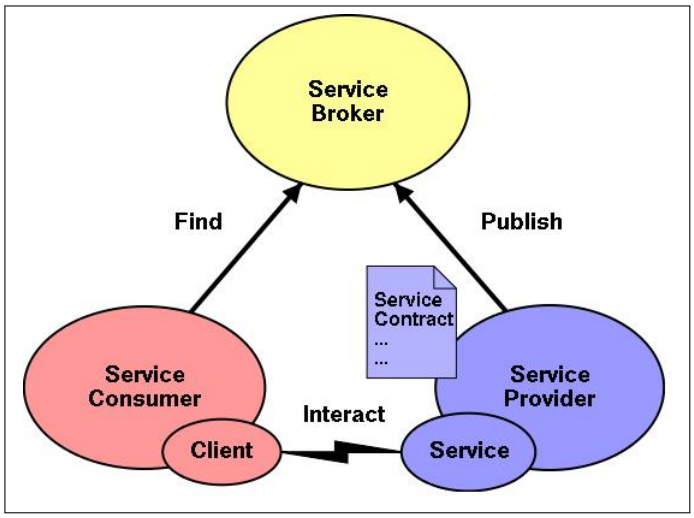


Abbildung 2.1.: Das Konzept von SOA
Quelle: [Haas 2003]

2.1.3. Erfolgskonzept und Vorteile einer SOA

Der Gegenstand einer SOA ist die Denkweise für ein Systemkonzept, die nur durch richtiges theoretisches Verständnis erfolgreich eingesetzt werden kann. Außerdem müssen, gemäß Jeffrey Walker, dem Gründer von TenFold Corporation [SoaWorld Juni 2007], spezifische technische Grundvoraussetzungen gegeben sein:

> *"Successful SOA strategies require an agile development platform, a robust and flexible applications infrastructure, and the empowerment of the right resources - those who understand the business processes - to build, manage, and continually improve that applications infrastructure."*

Die lose gekoppelten Services ermöglichen eine dynamische Reaktion und Anpassung des Systems. Je nach Anforderung können Services gebunden werden. Funktionen und Schnittstellen werden mit ihrer kleinstmöglichen Aufgabe, Bedeutung und Wertebereich definiert. Die Dienste sollen autonom sein, damit sie auch mit anderen Services zu einer neuen Einheite fungieren können. Durch diese Abstraktion und Unabhängigkeit ist auch ein hoher Grad an Wiederverwendbarkeit geschaffen, indem ähnliche Funktionsweisen nur einmal implementiert werden müssen und darauf folgend jedem anderen Service zur Verfügung stehen. Aufwand und Kosten können

gesenkt werden, indem unter Anderem durch diese Architektur die Wartbarkeit vereinfacht wird.

2.1.4. Die Infrastruktur von SOA - der Enterprise-Service-Bus (ESB)

Eine SOA ist kein festgelegtes System und besitzt keine vordefinierte Infrastruktur, sondern ist ein Systemparadigma. ESB ist eine Softwareeinheit, die eine SOA als Infrastruktur unterstützen kann. Es wird beispielsweise die Kommunikationssteuerung übernommen, damit sich die Services und deren Nutzer an den Bus koppeln können. In Folge dessen wird die Server-Client Direktverbindung ersetzt. Dabei ist es unerheblich, welche Technologien oder Protokolle von dem Service oder Consumer verwendet werden. ESB bietet die Möglichkeit einer Integration von verteilten Systemen, Applikationen, Unternehmen und Netzwerken und deren Interoperabilität, unabhängig von der darunter liegenden Software und den Standards.

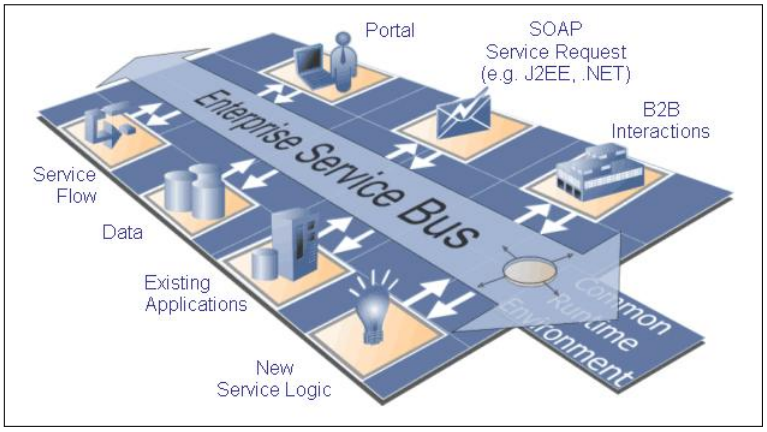


Abbildung 2.2.: Der Enterprise-Service-Bus
Quelle: [Woolf 2006]

Als bedeutende und unterstützte Funktionalitäten des ESB lassen sich folgende nennen:

- Konnektivität zu Elementen herstellen
- Verwalten und Binden von Applikationen und Services
- Routing von Nachrichten
- Transformationen und Komprimierung von Daten

Dabei ist der ESB selber serviceorientiert, sodass die Funktionalitäten Services sind [Vgl.: Masak 2007, S. 146]. Im Folgenden wird auf die Funktionalitäten detaillierter eingegangen.

Aufbau und Funktionsweise eines ESB

> *"Die wichtigste Aufgabe des ESB besteht darin, Interoperabiltät zu schaffen."*[Josuttis 2008, S. 64]

Um die Aufgabe der Interoperabilität umzusetzen, muss zum Einen erst einmal die Möglichkeit gegeben sein, dass zum ESB eine Konnektivität hergestellt werden kann. Deshalb muss der ESB geeignete Schnittstellen und Adaptoren bereitstellen. Ist eine Verbindung hergestellt, sollen Daten übertragen werden können. Dafür müssen die Nachrichten in ein entsprechendes Format transformiert werden. Die Verbindungsstellen unterliegen keiner vorherbestimmten Technologie, sondern die Heterogenität wird unterstützt. Damit die Nachrichten beim Empfänger auch schnell und sicher ankommen, stellt der ESB Funktionalitäten für ein intelligentes Routing bereit. Des Weiteren besteht die Möglichkeit im ESB Prozesse und ihre Aktivitäten zu analysieren und modellieren. Hinsichtlich des Arbeitsflusses und Datenaufkommens können an dieser Stelle Kontrollen und Überwachungen vorgenommen werden. Zudem kann der ESB als eine Art Debugger genutzt werden, in dem die verteilten Geschäftsprozesse protokolliert werden können. Dazu werden Informationen der Kommunikation dotiert, beispielsweise der genutzte Service, Aufrufer und die Laufzeit.

2.2. Web Service

Der Begriff Service stammt aus dem Englischen und steht für Dienstleistung. Es ist eine Tätigkeit oder Aufgabe, die von jemandem oder maschinell in Form einer Leistung für jemanden erbracht wird. In der heutigen Zeit werden Services vorab fälschlicher Weise mit der modernen Technik in Zusammenhang gesetzt. Nachstehend ein Beispiel [Josuttis 2008, S. 6]:

> *"Als Service wird eine zusammengehörige Menge von vermarktbaren Diensten bezeichnet, die einem autorisierten Nutzerkreis unter Nutzung standardisierter Kommunikationsprotokolle über wohldefinierte Schnittstellen angeboten werden. Services umfassen Schnittstellenbeschreibungen der von ihnen angebotenen Dienste, die in einer standardisierten Beschreibungssprache verfasst sind."*

Gegenstand dieser Literaturquelle ist der Versuch Services aus technischer Sichtweise zu beschreiben, mit Web Service Technologie und dessen Aufbau und Funktionsweise. In Folge dessen werden in Kapitel 2.2.1 Web Service anhand dieser Definition erläutert und identifiziert. Dabei gilt es einen Service getrennt von technischer Anschauungsweise zu betrachten, aus softwaretechnischer Sicht und als Gesellschaftsphänomen. Services sind allgegenwärtig anzufinden. In unserer Gesellschaft entstanden aus der Spezifizierung und Differenzierung von Dienstleistungen

die entsprechenden Berufszweige. Mit der folgenden Definition werden Services sowohl technisch als auch gesellschaftlich erfasst:

> Als Service ist eine spezifische Leistung gemeint, welche durch wohl definierte Aufgaben und Tätigkeiten gekennzeichnet ist und durch den Dienstleister erbracht wird. Zwischen Dienstleister und -nutzer gilt eine individuelle Regelung.

In der heutigen Zeit spielt die Serviceorientierung in der Softwaretechnik eine sehr wichtige Rolle. Es gilt als ein förderndes Paradigma für die Flexibilität und Agilität. Softwarefunktionalitäten in "kleine Bausteine" zu entwickeln, also die Services, bringt viele Vorteile mit sich. Derartige entworfene Software kann aus vielen Bausteinen zusammengesetzt werden, sodass eine Applikation nicht komplett neu entwickelt werden muss und Funktionalitäten[2] wiederverwendet werden. Daraus resultiert ein schnelles Wachstum an neuer Software, indem die Bausteine völlig unterschiedlich kombiniert werden können [Vgl.: Masak 2007, S. 4].

2.2.1. Was ist ein Web Service?

Als einen Web Service wird die Sammlung von Standards bezeichnet, die in verteilten Systemen Interoperabilität sicherstellen sollen. Die Standards legen die Bereitstellung von Diensten über eine Netzwerkressource in Kommunikations- und Übertragungsprotokollen und Beschreibungen der Schnittstelle fest. Ein Service ist folglich über das Internet, zum Beispiel mit einer bestimmten URI erreichbar. Ein Web Service ist ein in sich abgeschlossenes, vollständiges und plattformunabhängiges Element, das eine Geschäftsfunktionalität repräsentiert und sie über das Web zur Verfügung stellt. Die in Kapitel 2.2 genannte Definition von Service am Beispiel von Web Services, lässt sich nun sich folgend auf Web Services projizieren, indem die drei Eckpfeiler identifiziert werden können:

- **"Unter Nutzung standardisierter Kommunikationsprotokolle"– SOAP**
  Simple Object Access Protocol (SOAP) ist das Protokoll, das die Kommunikation und den Nachrichtenaustausch formal festlegt. Es basiert auf XML und HTTP.
- **"über wohldefinierte Schnittstellen angeboten" – UDDI**
  Universal Description, Discovery and Integration (UDDI) Standard für das Bekanntgeben und Finden von Services. Es fungiert als Verzeichnisdienst.
- **"in einer standardisierten Beschreibungssprache verfasst sind" – WSDL**

[2]Funktionalitäten können als Bausteine angesehen werden

Web Service Definition Language (WSDL) ist das Format für die Beschreibung von Schnittstellen der Web Services. Beschrieben wird unter Anderem, über welchen Uniform Resource Locator (URL) der Web Service zu erreichen ist und welche Methoden zur Verfügung gestellt werden.

Die Zusammenhänge dieser Standards und die allgemeine Funktionsweise von Web Services wird in Kapitel 2.2.3 ausführlich beschrieben.

2.2.2. Das Web Service Fundament – XML

Die eXtensible Markup Language (XML) ist der Standard des W3C für Datenaustausch im Web. Zudem legt die Metasprache das Fundament für Web Services.
Der Aufbau eines XML-Dokuments kann über Document Type Definition (DTD) oder XML Schema (XSD) strukturiert angegeben werden. Wird diese Struktur vom XML-Dokument erfüllt, ist es gültig. Wohlgeformt ist das Dokument, wenn es die W3C Empfehlungen einhält. Die Darstellung von XML-Dokumenten wird durch eXtensible Stylesheet Language (XSL) ermöglicht und die Transformation, beispielsweise zu Hypertext Markup Language (HTML) oder Portable Document Format (PDF) durch XSLT.

2.2.3. Funktionsweisen und Zusammenwirken von Web Services Standards

Damit ein Web Service genutzt werden kann, muss zuerst eine Beschreibung mit WSDL bereitgestellt werden. Der Dienstanbieter veröffentlicht daraufhin seinen Service mit der Schnittstellenbeschreibung im UDDI-Register. Daraufhin wird der Service für Konsumenten auffindbar und benutzbar. Die Registrierung und die Dokumentenübertragung geschehen hier nach dem Kommunikationsprotokoll SOAP. Über UDDI können Web Services gefunden werden, und zudem werden im UDDI-Register Informationen über den Provider zur Verfügung gestellt. Durch die veröffentlichte WSDL weiß der User, wie er den Web Service binden und benutzen kann. Durch die zusätzlichen Informationen kennt der Nutzer die Beschreibung des Bereitstellers und die Kontaktinformationen. Der Nachrichtenaustausch, sowohl von User zu UDDI und von User zum Web Service erfolgt über SOAP. Dadurch ist der Nachrichtenaustausch und Kommunikationsablauf genau vorherbestimmt. Details zu diesen Technologien werden in den folgenden Kapiteln genannt.

2.2.4. Allgemeines über SOAP als Kommunikationsprotokoll

Das Simple Object Access Protocol (SOAP) ist ein auf XML basierendes Protokoll, das für die Kommunikation, also den Austausch von Informationen, zuständig ist. Es liegt in der Version 1.2 vor, die vom W3C am 27.April 2007 als Empfehlung

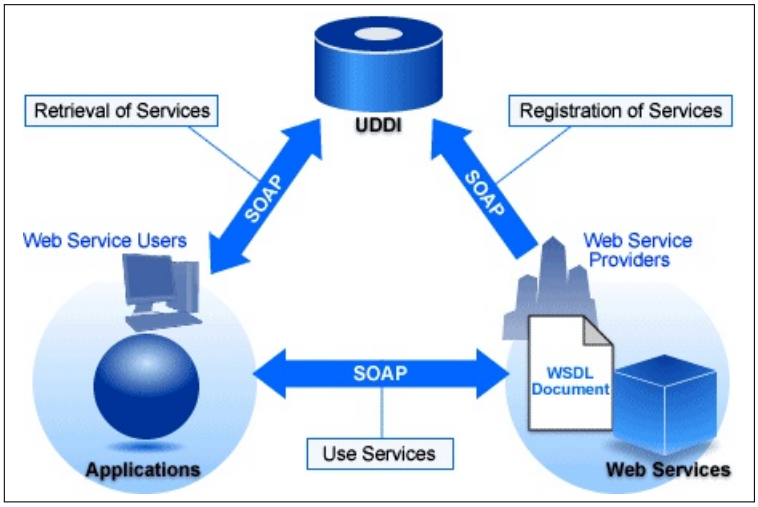


Abbildung 2.3.: Zusammenwirken UDDI, SOAP und WSDL
Quelle: [Manhart März 2007]

verabschiedet wurde [Gudgin u. a. April 2007].
In SOAP wird beispielsweise festgelegt, welcher Kommunikationskanal verwendet wird und wie Daten komprimiert werden. Das Protokoll legt Regeln für die Kommunikationsabläufe fest.

Aufbau

Im Folgenden wird der Aufbau einer SOAP-Nachricht beschrieben. In Abbildung 2.4 wird die Grundstruktur gezeigt.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header>...</env:Header>
<env:Body>...</env:Body>
</env:Envelope>
```

Abbildung 2.4.: Aufbau von SOAP

SOAP Envelope
Envelope (Briefumschlag) enthält die Informationen der Nachricht. Diese werden in den Unterelementen Header und Body angegeben. Die in *xmlns:env* referenzierende URI legt die verwendete SOAP-Version und die damit vorgegebene Schema-Definition fest.
SOAP Header
Bei dem Header handelt es sich um ein optionales Element. Wenn es bei einer SOAP-Nachricht mitverwendet werden soll, muss es explizit als erstes Unterele-

ment der SOAP-Envelope angegeben sein. Das Header-Element umfasst Routing-Informationen, die für eine Zwischenstation auf dem Weg zum SOAP-Empfänger bestimmt sein können [Vgl.: Thiele und Habich 2007, S. 23].

SOAP Body

Der Body ist ein Pflichtelement, in dem die eigentlichen Informationen der Nachricht enthalten sind. Ist im Body-Element ein Unterelement Fault verzeichnet, ist bei der Nachrichtenumsetzung ein Fehler aufgetreten. Dieses Unterelement verfügt des Weiteren über ein Code- und Reason-Element, die genauere Informationen über den Fehler angeben.

2.2.5. Definition eines Web Services mit WSDL

Web Service Description Language (WSDL) ermöglicht die Beschreibung von Web Services unabhängig von dem verwendeten Protokoll. Es liegt in der Version 2.0 seit Juni 2007 als Standard von W3C vor [Ryman u. a. Juni 2007]. Die Sprache gilt als eine Schnittstellenbeschreibung. WSDL ist essentiell für Provider, um Schnittstellen von Web Services zu definieren und den Dienst anzubieten, beziehungsweise zu veröffentlichen. Diese Publizierung wird üblicherweise mit UDDI vollzogen, welches als eine Art Register (Verzeichnis) fungiert. Dabei basiert WSDL auf XML und stellt damit die Informationen eines Web Service dar, wie beispielsweise die Schnittstellen, Operationen und Funktionen und deren Adresse. In Abbildung 2.5 ist die Grundstruktur eines WSDL-Dokumentes mit deren Hauptkomponenten abgebildet. In den folgenden Kapiteln wird die Struktur beschrieben, wie WSDL auf Basis von

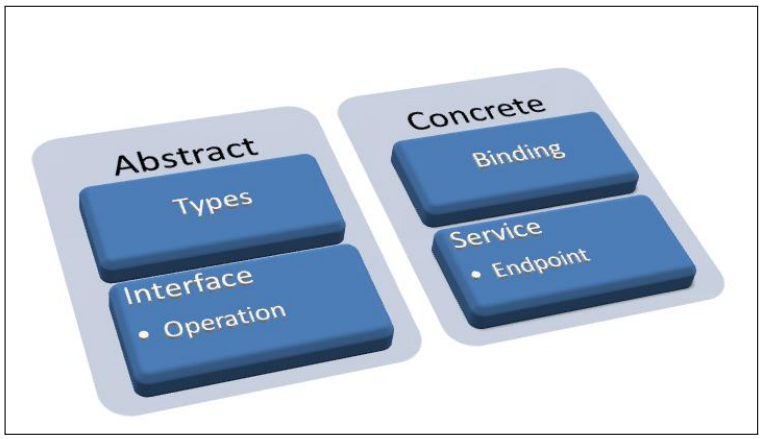


Abbildung 2.5.: Grundstruktur von WSDL

XML im Detail aufgebaut ist. Dabei wird der Aufbau einer WSDL in einen abstrakten und einen konkreten Teil gegliedert. Durch diese Trennung ist es möglich den Web Service und seine Funktionen auf abstrakter Schicht zu beschreiben und

die technischen Gegebenheiten, wie Protokolle und Netzwerkadressen, zuerst außen vor zu lassen.

Aufbau einer WSDL – abstrakter Teil

In dem abstrakten Teil einer WSDL werden durch die Elemente types, message und portType angebotene Operationen und Nachrichtentypen beschrieben.

- **types** – in dem Element **types** erfolgt eine Beschreibung der Datentypen, die beim Datenaustausch verwendet werden. Es wird empfohlen die Angabe mit dem XML-Schema zu machen. In anderen Teilen der WSDL kann auf diese Datentyp-Definitionen zugegriffen werden um In- und Outputparameter festzulegen.
- In der Abbildung 2.6 ist der Aufbau des **types** dargestellt.

```
...
<types>
<xs:complexType name="AutoTyp">
<xs:sequence>
<xs:element name="Besitzer" type="xs:string" />
<xs:element name="Kilometer" type="xs:double" />
</xs:sequence>
</xs:complexType>
</types>
...
```

Abbildung 2.6.: Struktur des WSDL-Elements **types**

- **interface** – abstrakte Definition der Nachrichten, die ausgetauscht werden. Es werden die angebotenen Operationen in der **operation**-Komponente definiert und beschrieben. Zudem kann eine beliebige Anzahl von **fault** Fehlerkomponenten im **interface** angeben sein. Des Weiteren kann auf andere Schnittstellen verwiesen werden, genauso können andere Komponenten auf das **interface** referenzieren.
  - **operation** – über diese definierten Methoden kann ein User mit dem Web Service in Wechselwirkung treten, die Operationen können referenziert und benutzt werden. Es ist erforderlich bei einer Operation Message Exchange Patterns (MEP) beim Attribut **pattern** anzugeben. Es zeigt an, dass diese MEP-Referenz vom Service verwendet wird [Vgl.: Dostal u. a. 2005, S. 92].

  In der Abbildung 2.7 ist der Aufbau des **interface** dargestellt.

Aufbau einer WSDL – konkreter Teil

In dem konkreten Teil einer WSDL werden durch die Elemente binding und service die Adressierung und Bindung von Protokollen eines Services beschrieben.

```
...
<interface name="AutoInterface">
<fault name="bremsAusnahme" element="bsns:bremsenUngültig"/>
<operation name="gasGeben" pattern="URItoRef"> ... </operation>
<operation name="gangSchalten" pattern="URItoRef">...</operation>
<operation name="bremsen" pattern="URItoRef">...</operation>
</interface>
...
```

Abbildung 2.7.: Struktur des WSDL-Elements **interface**

- **binding** – beschreibt die technische Abbildung der abstrakt definierten Operationen und Nachrichten. Dies geschieht durch Angabe und durch Festlegung des zu verwendenden Protokolls. Die Syntax von **binding** wird in Abbildung 2.8 beschrieben.

```
...
<description>
<binding name="SOAPBinding" interface="AutoInterface" type="URItoSOAPschema">
...
</binding>
</description>
...
```

Abbildung 2.8.: Struktur des WSDL-Elements **binding**

- **service** – in diesem Element kann eine Menge von **endpoint**-Elementen angegeben werden, die an (physikalisch) unterschiedlichen Orten, an Schnittstellen des Service angeboten werden [Vgl.: Dostal u. a. 2005, S. 99].
  - **endpoint** – spezifiziert den physikalischen Endpunkt, an dem der Service verfügbar ist. Dieses Element ist vorgeschrieben, nur so sind ein Service und seine Schnittstellen aufrufbar.

  Folgende Abbildung 2.9 veranschaulicht den Aufbau von **service**

```
...
<description>
<service name="AutoSerive" interface="AutoInterface">
<endpoint name="AutoEntpunkt" binding="SOAPBinding"
address="http://www.BeispielURI.com/AutoService">
</endpoint>
...
</service>
</description>
...
```

Abbildung 2.9.: Struktur des WSDL-Elements **service**

2.2.6. Verzeichnisdienst und Register UDDI

Damit Web Services in verteilten Systemen und Netzwerken, wie die weltweite Verteilung im Internet, auffindbar und nutzbar sind, benötigt es eine Instanz, die wie

ein Verzeichnisdienst wirkt. Diese Aufgabe übernimmt Universal Description, Discovery and Integration (UDDI). UDDI ist ein Register, in dem Serviceprovider ihre Dienste mit essenziellen Informationen eintragen, die dem Konsumenten das Finden und Binden des Dienstes realisierbar machen. Es ist genau festgelegt, welche Informationen benötigt und eingetragen werden, dabei basiert UDDI auf SOAP und XML.
Durch diesen standardisierten Verzeichnisdienst wird das Veröffentlichen und Verwenden von Web Service schneller, leichter und dynamischer. Ein UDDI-Register kann entweder über den Web Browser oder elektronisch über spezifische Schnittstellen der UDDI-API (Aplication Programming Interface) angewendet werden.

Aufbau und Datenstruktur von UDDI

Informationen, die einen Service und seinen Provider auszeichnen, werden bei UDDI in drei Kategorien aufgeteilt:

- **White Pages** – White Pages beinhalten all die Informationen, die einen Serviceanbieter kennzeichnen und den Kontakt ermöglichen, beispielsweise Adresse, Name und Telefonnummer.
- **Yellow Pages** – in dieser Struktur werden Informationen über die Provider nach deren Branche, also Aufgaben- und Tätigkeitsbereichen, kategorisiert.
- **Green Pages** – diese Struktur umfasst die technische Beschreibung des Services. Die Adresse, wie die URL, die das Finden und Nutzen des Services ermöglicht, sowie die Schnittstellenbeschreibung des WSDL-Dokuments.

Diese Kategorien von Informationen über Service und Serviceprovider werden über eine auf XML basierende standardisierte Datenstruktur ausgetauscht:

- businessEntity – das Hauptelement, das Informationen über die Firma beinhaltet, wie beispielsweise Name und Kontaktinformationen, wie Telefonnummer und Email. Ein businessEntity beinhaltet businessService.
- businessService – umfasst die Beschreibung von Services. Die Services werden in Kategorien gruppiert und darüber Informationen bereitgestellt, zum Beispiel der Gruppenname. Die Struktur enthält ein oder mehrere bindingTemplates und tModels.
- bindingTemplate – diese Struktur beinhaltet die technische Beschreibung eines Web Services und dessen Eigenschaften und Besonderheiten. Es stellt die Informationen bereit, die benötigt werden um einen bestimmten Service aufzurufen und zu binden, wie die URL, über die der Web Service erreichbar ist. bindingTemplates werden durch ein angebundenes tModel näher erläutert. Ein bindingTemplate kann nur zu einem businessService gehören.

- tModel – spezifiziert die technische Beschreibung, unter anderem durch das veröffentlichete WSDL-Dokument des Providers werden die meisten Informationen gewonnen.

2.2.7. Komposition von Services - Orchestrierung und Choreographie

Um in verteilten Systemen Prozesse mit Web Services zu realisieren, werden Operationen benötigt, die das Zusammenfügen mehrerer Prozesse zu einer Anwendung ermöglichen. Dienstkomposition meint die intelligente Verknüpfung von Web Services zu einer zusammenhängenden Einheit. Die Services werden kombiniert und können von den Funktionalitäten eines anderen Services Gebrauch machen oder eigene Ergebnisse einem anderen Dienst zur Verfügung stellen. Es existieren drei Kompositionstypen, nach denen das Zusammenwirken von komponierten Services festgelegt ist [Masak 2007, S. 284 f.]:

- Horizontal – Services werden nacheinander ausgeführt und Nachfolgern werden Ergebnisse zur Verfügung gestellt.
- Vertikal – Services erbringen ihre Aufgabe, indem sie andere Services dafür aufrufen und verwenden.
- Hybrid – Kombination von horizontaler und vertikaler Kompositionsart.

Es werden zwei Arten der Dienstkompositionen unterschieden, die Orchestrierung und die Choreographie. Die Unterschiede der beiden Kompositionsarten werden in Abbildung 2.10 veranschaulicht. Choreographie beschäftigt sich mit der Reihenfolge

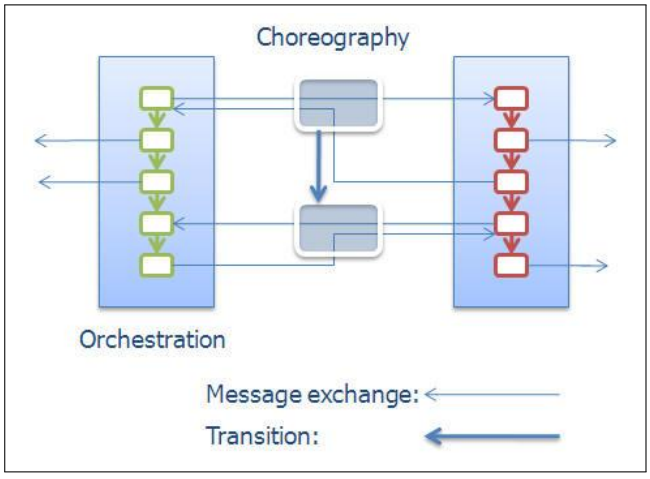


Abbildung 2.10.: Unterschiede von Choreographie und Orchestrierung
Quelle: http://www.ebpml.org/blog/imgB1.jpg (Letzter Zugriff Juli 2009)

von Serviceaufrufen und Nachrichtenabfolgen. Und wird laut Glossar des W3C [Haas und Brown 2004] folglich beschrieben:

> *"A choreography defines the sequence and conditions under which multiple cooperating independent agents exchange messages in order to perform a task to achieve a goal state."*

Laut Glossar des W3C ist die Choreographie ein Mittel, um die Reihenfolge und Bedingungen für den Nachrichtenaustausch mehrerer und unabhängiger Verbindungsstellen zu definieren. Die Kommunikation erfolgt um ein erwünschtes Ergebnis einer Aufgabe zu erreichen. Die Choreographie stellt lediglich die Reihenfolgen und den Nachrichtenfluss fest. Dies geschieht aus globaler Sicht für jeden Dienst.

> *"Hierbei kontrolliert eine zentrale Kontrollinstanz das Zusammenspiel der für die Ausführung eines Prozesses erforderlichen Services."* [Kupsch 2006, S. 86]

Die Beschreibung von Kommunikationsabfolgen und die Kontrolle der Services erfolgt durch eine globale Instanz, die vergleichbar ist mit einem Koordinator, der dafür sorgt, dass die einzelnen Verbindungsinstanzen und deren Anwendungen zusammenwirken. Dabei bleiben die Prozesslogiken und die Bedingungen außen vor. Somit können beispielsweise keine Zustände festgelegt werden, unter denen ein Nachrichtenaustausch erst stattfinden soll. In der Literatur [Kupsch 2006, S. 86] wird die Komposition der Orchestriertung im Vergleich beschrieben:

> *"Eine Orchestrierung hingegen beschreibt die Prozesslogik innerhalb eines Geschäftsprozesses aus Sicht eines einzigen Teilnehmers durch die Aufrufreihenfolge interner und externer Web Service."*

Orchestrierung stellt somit die Aktivitätenabläufe eines Services detailliert dar. Dabei wird vor allem auf die Prozesslogik eingegangen. Es können Bedingungen, Alternativen und Abhängigkeiten zwischen Instanzen und deren Nachrichtenaustausch festgelegt werden. Dabei wird die Kommunikation immer aus der Sicht eines Services gesehen. Im Vergleich zur Choreographie, bei der aus globaler Sicht dokumentiert wird, wird bei der Orchestrierung eine lokale Beschreibung vorgenommen.

2.2.8. Orchestrierung mit WS-BPEL

Die Web Service Business Process Execution Language liegt seit April 2007 in der Version 2.0 als standardisierte Technologie von OASIS vor [Oasis 2007]. WS-BPEL 2.0 ist eine Erweiterung, beziehungsweise der Nachfolger von BPEL4WS, welches ein Zusammenschluss von den Technologien XLANG[3] und WSFL[4] ist.

[3] eine Entwicklung von Microsoft
[4] eine Entwicklung von IBM

WS-BPEL ermöglicht die Ablaufsteuerung von Geschäftsprozessen, die mit ihren Funktionalitäten auf Web Service abgebildet werden, als strukturierte Aktivitäten. Dabei können Bedingungen und Abhängigkeiten von Ergebnissen anderer Prozesse definiert werden. WS-BPEL basiert auf XML und lässt sich vereinfacht in Web Services integrieren. Dabei enthält die Syntax von WS-BPEL speziell Elemente, die für aufwendige Prozesse zugeschnitten sind und auch mit Parallelität und langen Laufzeiten gut umzugehen wissen.

Konstruktionen von WS-BPEL

In WS-BPEL werden Elemente angeboten, die die Teilnehmer an Aktivitäten und deren Zuständigkeiten definieren lassen (*partner* und *partnerLink*). Durch das Element *variables* lassen sich Parameter und Variablen definieren. Es sind Werte erforderlich, um einen Nachrichtenfluss erfolgen zu lassen, oder Variablen, die von Partnern genutzt werden. Variablen ermöglichen die Kontrolle und Steuerung in Abhängigkeit von Bedingungen und Zuständen und spezifizieren das Charakteristikum eines Prozesses. Die Konstrukte *faultHandler*, *eventHandler* und *compensationHandler* dienen zur Handhabung und Behandlung von Fehlern und Events. Für die Steuerung und Kontrolle von Aktivitäten stehen in WS-BPEL folgende Elemente zur Verfügung:

- ***receive*, *reply*, *invoke*** – Konstrukte zum Angeben des Kommunikationsstatus, wie beispielsweise receive für Empfangen.
- ***throw*, *rethrow*, *exit*, *compensate*, *compensateScope*** – mit diesen Elementen können entstandene Fehler behandelt werden, indem darauf reagiert wird.
- ***flow*, *sequence*, *pick*, *scope*, *extensionActivity*** – durch diese Elemente können Strukturen von Abläufen gebildet werden.
- ***wait*, *empty*, *validate*, *assign*** – Konstrukte, die dazu verwendet werden, die anderen Elemente zu unterstützen. Zum Beispiel lässt sich mit *wait* in parallelen Abläufen, die durch *flow* erzeugt werden, ein Wartezustand einer Aktivität realisieren.
- ***if*, *else*, *elseif*** – bedingungsabhängige Durchführung eines Blocks von Aktivitäten, der durch diese Elemente eingegrenzt ist.
- ***while*, *forEach*, *repeatUntil*** – bedingungsabhängige Wiederholung von Aktivitäten. *forEach* unterstützt sowohl sequentielle als auch parallele Durchführungen von Aktivitäten.

2.2.9. Choreographie mit WS-CDL

Web Service Choreography Describtion Language (WS-CDL) liegt seit 2005 vom W3C als "Recommendation Candidate" für die Standardisierung vor [Kavantzas u. a. 2004]. Mit WS-CDL lassen sich Kommunikationsabläufe unterschiedlicher Teilnehmer bestimmen. Dabei wirken die Teilnehmer zusammen zum Erledigen einer Aufgabe. Die Choreographie Sprache basiert auf XML. In der Abbildung 2.11 wird der Aufbau eines WS-CDL Dokuments gezeigt.

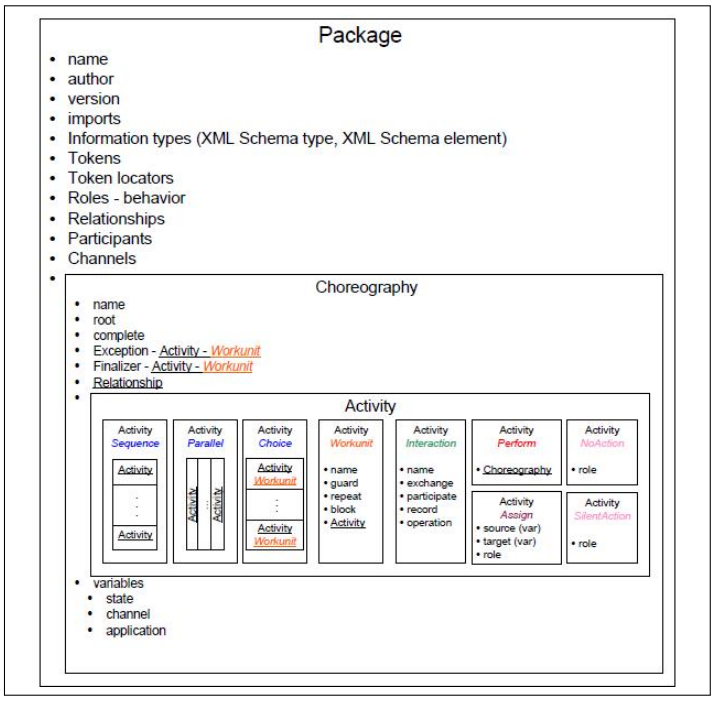


Abbildung 2.11.: Struktur der Choreographiesprache WS-CDL
Quelle: [Barros u. a. März 2005, S. 7]

Das *package*- Element ist das Hauptelement eines WS-CDL Dokuments. Es umfasst beispielsweise eine Sammlung von Typen (Verbindungsstellen) und deren Rollen und Zusammenwirken, sowie die choreographische Beschreibung. An dieser Stelle wird nicht weiter auf WS-CDL eingegangen, da die Choreographie für die Optimierung von datenintensiven Prozessen in SOA für dieses Projekt nicht von wesentlicher Bedeutung ist. Diese Methodik beschreibt ausschließlich den Nachrichtenablauf unterschiedlicher Dienste und deren Zusammenwirken. Genauso unterstützt die Choreographie Sprache nicht die Steuerung und Kontrolle von Aktivitäten der Prozesse. Dieses Management der Aktivitäten ist aber von signifikanter Bedeutung, wenn es darum geht die Prozessaktivitäten und den daraus resultierenden Datenfluss zu optimieren. Weitere Informationen zu WS-CDL können unter Anderem auf

der Webseite des W3C entnommen werden[5].

2.3. Management von Geschäftsprozessen

Ein Geschäftsprozess ist eine Menge von Funktionalitäten und Aktivitäten, die dem Unternehmen unterstützend hinzukommen, in dem sie zur Erreichung der Unternehmensziele beitragen. Aus softwaretechnischer Sicht wird ein Geschäftsprozess folgendermaßen beschrieben [Masak 2007, S. 176]:

> *"Im Kontext der Softwareentwicklung wird in der Regel unter einem Geschäftsprozess die inhaltlich abgeschlossene, zeitlich-sachlogische Abfolge von Funktionen verstanden, die zur Bearbeitung eines für die Leistungserbringung der Organisation relevanten Objekts erforderlich ist."*

Ein Geschäftsprozess wird aus technischer Sichtweise auch als Workflow[6] bezeichnet.

2.3.1. Lebenszyklus eines Prozesses

Der Prozesslebenszyklus definiert die Phasen eines Prozesses, die von der Vorbereitung und Beschreibung, bis hin zur Terminierung und Aussonderung oder Wiederbeziehungsweise Weiterverwendung durchlaufen werden. Das Wort Zyklus stammt von dem lateinischem cyclus[7]. Abstrakt gesehen werden die Phasen in vorbestimmter Reihenfolge kreisförmig durchlaufen. Das Muster der Reihenfolge tritt wiederkehrend auf. Bei dem Prozesslebenszyklus müssen nicht alle Phasen durchlaufen werden, optional je nach Anforderung können sie angepasst und verwendet werden. Der charakteristische Kreislauf ist in Abbildung 2.12 dargestellt. (1) Zuerst wird ein Prozess geplant. Die Funktionen werden entworfen, die benötigt werden um Aufgaben zu erledigen. Es wird ein genaues Konzept des Prozesses und seiner Abläufe und Aktivitäten ausgearbeitet. Der genaue Prozesszweck und Anforderungen werden festgelegt. (2) Anhand des Entwurfes und Konzeptes kann der Prozess entwickelt werden. (3) Nach der Entwicklung kann der Prozess genutzt werden. Er kann nun dem Unternehmenszweck mit seinen Bearbeitungen und bereitgestellten Funktionalitäten dienen. (4) Ist die Ausführung vollendet, kann der Prozess beendet werden. Werden die Arbeitsschritte nicht noch einmal benötigt, wird der Prozess entfernt. Andernfalls kann der Prozess wiederverwendet werden oder ein Kindprozess erzeugt werden (Vererbung).

[5] siehe [Kavantzas u. a. 2004]
[6] im Deutschen auch Arbeitsfluss
[7] deutsch Kreis

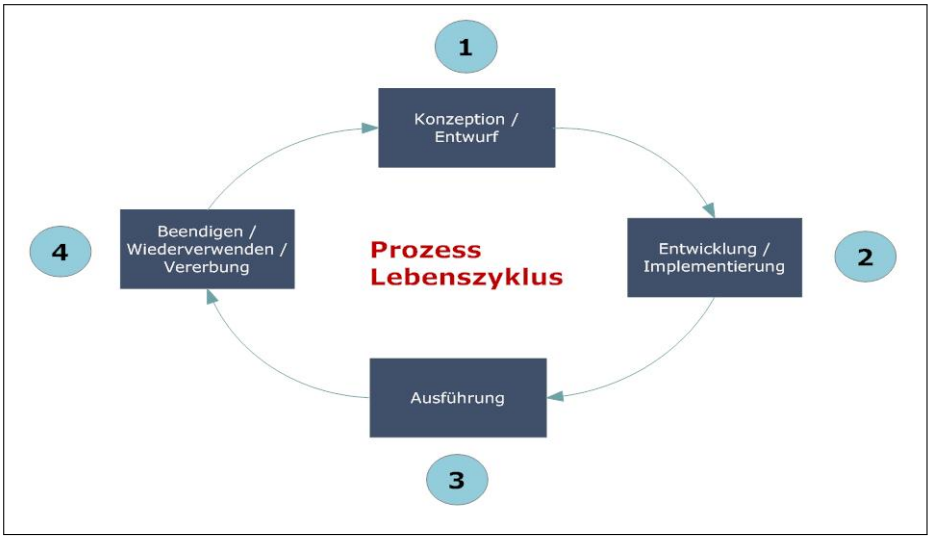


Abbildung 2.12.: Der Lebenszyklus eines Prozesses

2.3.2. Business Process Management (BPM)

Das Business Process Management (BPM) beschreibt Vorgehensweisen und Methoden für die Analyse, Verwaltung, Kontrolle und Verbesserung von Geschäftsprozessen. BPM zeichnet aus, dass wiederholende und prägende Abläufe in einem Unternehmen erkannt und daraus Geschäftsprozesse definiert und modelliert werden. Anhand dieser Beschreibung werden Verbesserungen und Optimierungen vorgenommen. Zudem ermöglicht die Modellierung eine Bobachtung und Kontrolle von Prozessen. Diesbezüglich nimmt Wolfgang Weigend, der im Management der BEA GmbH tätig ist, Stellung [Weigend September 2006]:

> *"BPM bietet einen hohen Abstraktionsgrad zur Definition von Geschäftsprozessen sowie wichtige Fähigkeiten für das Monitoring und Management dieser Prozesse. Services liefern die erforderlichen Anwendungsfunktionen, um diese Prozesse zu unterstützen. Eine SOA schließlich verbindet die Services miteinander und bildet so die Basis für ein agiles und flexibles Unternehmen."*

2.3.3. Phasen des BPM

In Abbildung 2.13 werden die Phasen von BPM dargestellt, die zyklisch durchlaufen werden können, also in einem wiederkehrenden Muster.
Analyse: In der Analysephase wird der Anwendungsbereich des Prozesses festgelegt. Diesbezüglich werden zu dem Bereich auch der Zweck des Prozesses und die Anforderungen bestimmt. Es erfolgt eine genaue Analyse des Prozesses und seiner Anforderungen. Dabei werden die einzelnen benötigten Aktivitäten definiert. Es

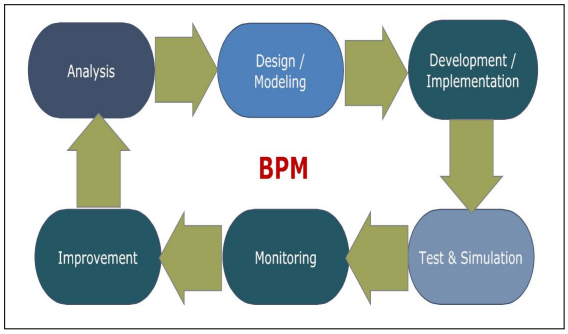


Abbildung 2.13.: Die Phasen des Business Process Managements

werden bestehende Prozesse ausgewertet (Ist-Analyse) und Ziele beschrieben, die durch den Prozess erreicht werden sollen.
Design/Modeling: Es folgt der Entwurf eines Konzeptes für den Prozess und der Reihenfolge seiner Aktivitäten. Desweiteren können verwendete Daten, Informationen und Zustände beschrieben werden. Die genaue Beschreibung und Spezifizierung des Prozesses an dieser Stelle vereinfacht die Orchestrierung von Prozessen.
Development/Implementation: Auf Basis der Modellierung und Entwürfe wird der Prozess erstellt. Berücksichtigt werden müssen bei der Implementierung die genauen vorherbestimmten Abläufe und Zusammenhänge.
Test and Simulation: Nachdem der Prozess entworfen und entwickelt wurde, können Tests und Simulationen durchgeführt werden, anhand derer der Erfolg und die Korrektheit des Prozesses bewertet werden können.
Monitoring: Genaue Beobachtung und Kontrolle der Prozessausführung. Prozesse werden überwacht und hinsichtlich der festgelegten Ziele des Unternehmens bewertet. Die Kontrolle unterstützt auch die Prozessmanagement Strategie, indem wichtige Informationen hieraus gewonnen werden können. Hinsichtlich der Kontrolle und Überwachung lassen sich Strategien ableiten, wie zum Beispiel an welcher Stelle Optimierungsbedarf besteht.
Improvement: Verbesserung des Prozesses und seiner Abläufe. Diese Optimierung erfolgt auf Basis der Beobachtungen und Analysen.

2.3.4. Vorteile durch Prozessmanagement

Das Prozessmanagement hilft dabei einen besseren Überblick über den Arbeitsfluss zu erhalten. Dadurch können Überlastungen oder andere Probleme schneller lokalisiert und bei Beobachtung kontrolliert werden. Die Modellierung und Beschreibung der Prozesse erzeugen transparente Abläufe, die eine Effizienzsteigerung mit sich führen. Grund hierfür sind klar definierte Abläufe. Prozesse können hinsichtlich ihres Datenaufkommens und ihrer Laufzeit besser erfasst und optimiert werden. Dieses wird zum Einen durch eine Beobachtung und Kontrolle ermöglicht und zum

Anderen durch eine verbesserte Gestaltung der Prozesse. Auch wird die Performance und Produktivität optimiert, indem die Auswirkungen und Veränderungen durch Nutzung der Prozesse deutlich werden.

2.3.5. Unterstützende Technologien des BPM

Für einige Phasen des BPM (Kapitel 2.3.3) gibt es Technologien, die unterstützend eingesetzt werden können.
Für die Analyse von Prozessen können Brenchmarking und SWOT-Analyse hilfreich eingesetzt werden. Brenchmarking meint Vergleich und Analyse von eigenen Unternehmensprozessen mit Referenzwerten. Daraus lassen sich wichtige Informationen und Werte gewinnen, beispielsweise wie effektiv wirklich ein Prozess ist und an welchen Stellen Optimierungsbedarf besteht. Die SWOT-Analyse meint die Analyse von Stärken, Schwächen, Risiken und Chancen. Da diese Analyse überwiegend in betriebswirtschaftlichen Bereichen Anwendung findet, wird an dieser Stelle nicht weiter darauf eingegangen.
Dem Design und der Modellierung stehen einige Technologien als Unterstützung zur Verfügung. Unter Anderem kann BPEL für die Beschreibung verwendet werden. Mit BPEL erfolgt die Definierung durch Syntax. Auch Electronic Business using eXtensible Markup Language (ebXML) kann für die Modellierung von Zusammenhängen verwendet werden. Mit Business Process Modeling (BPM) und Business Process Modeling Notation (BPMN) lassen sich die Geschäftsprozesse grafisch darstellen. Für das Monitoring von Geschäftsprozessen kann Business Acitivity Monitoring (BAM) eingesetzt werden.

Business Process Execution Language (BPEL)

BPEL wurde schon im Detail im Kapitel 2.2.8 beschrieben. In diesem Zusammenhang wurde die Prozessbeschreibungssprache dazu verwendet Orchestrierung zu unterstützen. Doch auch im Prozess Management findet BPEL seinen Einsatz. Denn die Sprache ermöglicht die Dokumentation von Prozessabläufen Schritt für Schritt. Des Weiteren wird durch BPEL die Steuerung und Kontrolle von Geschäftsprozessen während der Laufzeit durchführbar. Die Abläufe können während der Laufzeit von Prozessen gesetzt und geändert werden.

Business Process Modeling Notation (BPMN)

BPMN ist eine Notation, nach der Geschäftsprozesse visualisiert und modelliert werden. Anhand von bereitgestellten Symboliken eignet sich BPMN vor allem für die grafische Darstellung von Prozessen und Workflows. Als Alternative für die grafische Darstellung ist Unified Modeling Language(UML) zu nennen. Im Vergleich zu BPMN ist UML aber nicht prozessorientiert.

Business Activity Monitoring (BAM)

Business Activity Monitoring liefert in Echtzeit Informationen über den Arbeitsfluss von Prozessen. Durch die Echtzeit-Prozessanalysen werden Problemstellen schnell identifiziert und Optimierungsbedarf erkannt. Dabei können Analysen auch für erwünschte Zeiträume in chronologischer Reihenfolge gespeichert werden. Dadurch werden eine Rückverfolgung und zeitlich[8] bedingte Auswertung möglich. Somit lassen sich an Wochenenden oder zu bestimmten Zeitpunkten bedingte Überlastungen erkennen. Es können dadurch entsprechende Vorkehrungen getroffen werden. Folgend ein Anwendungsbeispiel aus der Praxis [Microsoft 2009]: *"So kann die METRO Group heute bereits eine Stunde vorher sagen, ob sie in einem ihrer Supermärkte eine weitere Kasse öffnen muss, um ihren Kunden Wartezeiten zu ersparen."*

Electronic Business using eXtensible Markup Language (ebXML)

ebXML (Electronic Business using eXtensible Markup Language) bietet standardisierte Methoden an, die eine Dokumentation und automatische Durchführung des Informationaustausches zwischen Geschäftsprozessen und Kooperationspartnern ermöglichen. Detaillierte Informationen über Teilnehmer und Geschäftsprozesse können in einem Register gespeichert werden. Die Daten, wie Informationen, Beschreibungen und Nachrichten, basieren auf XML [Vgl.: Masak 2007, S. 256]. Durch die beschriebenen Konzepte von Web Service (Kapitel 2.2.3) und SOA (Kapitel 2.1.3) ist bekannt, dass Unternehmen ihre Prozesse anbieten müssen, in dem sie veröffentlicht und in einem Register (wie zum Beispiel UDDI) gespeichert werden. ebXML bietet ein Register an, das aber speziell für Business-Prozesse ausgelegt ist. In Diesem werden spezifische Informationen abgelegt, die auch Beschreibung zu Kooperationen und Partnern enthalten [Vgl.: Kipp 2006].

Workflow Management (WfM)

Ein Workflow beschreibt die Gesamtheit der unterschiedlichen Teile von Prozessen, also jede Folge von Aktivitäten [iXenso Software-Solutions e.K 2007]:

> *"Workflow betrachtet nicht nur einzelne Arbeitsvorgänge wie die Bestellannahme oder die Bestellweitergabe, sondern ganze Geschäftsvorfälle bzw. Arbeitsabläufe, d.h. die Bestellung von der Annahme bis zur Auslieferung."*

Das Workflow Management hat als Aufgabe die einzelnen Arbeitsschritte von Geschäftsprozessen zu automatisieren. Dabei werden die Arbeitsabläufe dokumentiert. Es werden vor allem immer wiederkehrende Abläufe identifiziert und automatisch

[8] zum Beispiel wochentags

gesteuert. Zuerst einmal definiert das Workflow Management System die Prozesse und erzeugt Instanzen. Danach werden Interoperabilität und Transaktionen kontrolliert.

2.4. Datenorganisation und -speicherung

Datenverwaltungssysteme stehen vor der Herausforderung, dass der Datenbestand in einer IT-Landschaft kontinuierlich historisch anwächst. Mit steigender Anzahl des Fundus nimmt der Umfang und die Komplexität enorm zu, sodass durch ein Verwaltungssystem den Nutzern die Verwendung der Daten erleichtert wird. Die richtige Datenorganisation schafft Ordnung und Übersichtlichkeit eines Bestandes. Vor allem Mengen an Informationen erfordern eine effektive Organisation, damit bei großen Ansammlungen auch gewünschte Informationen schnell wiederzufinden sind. Aus diesen Gründen entsteht die Motivation, Paradigmen und Technologien anzuwenden, die den Umgang mit hohem Datenbestand erleichtern.

2.4.1. Allgemeines über Partitionierung

Unter Partitionierung versteht man das physikalische Aufteilen von Datenverwaltungssystemen, um so gezielt die zu speichernden Volumen in Abhängigkeit von ihrer Logik, Beschaffenheit und der Verwendung getrennt ablegen zu können. Physikalisch kann durch Partitionierung eine Tabelle über ein Dateisystem verteilt werden. Die Tabelle wird nicht nur auf eine Datei gespeichert, sondern kann je nach Konfiguration des Administrators auf mehrere Dateien aufgegliedert werden. Die Tabelle bleibt aber logisch zusammenhängend, sodass die Partitionierung für den Nutzer verborgen bleibt. Dadurch kann die Tabelle nach festgelegten Regeln in einzelne Teile zerlegt werden. Es werden zwei Partitionierungsarten für Datenbanktabellen unterschieden:

- **Vertikale Partitionierung**
  Den Partitionen können verschiedene Spalten zugeordnet werden. Jede Partition kann unterschiedlich viele Spalten enthalten.
- **Horizontale Partitionierung**
  Den Partitionen können verschiedene Zeilen zugeordnet werden. Über eine vorherbestimmte Spalte wird die Zuweisung vollzogen.

Funktionen der Partitionierung

Für die Partitionierung von Datenbanktabellen stehen vier Methoden zur Verfügung. Jeder Methode muss zusätzlich ein Wert angegeben werden, nach dem die Aufteilung erfolgt. Der Wert muss ein Integer sein oder eine Funktion, die einen Integer zurückgibt.

- **RANGE-Partitionierung**
  Partitionen können nach Wertebereich aufgeteilt werden. In festgelegten Partitionen sind folglich nur Werte in vorbestimmten Bereichen vorhanden. Der Wertebereich ist Partition Key.

- ***LIST*-Partitionierung**
  Ähnelt der *RANGE*-Partitionierung insofern, dass es eine Art Bereichspartitionierung ist. Der Unterschied liegt darin, dass eine Liste von Bereichen angegeben wird und somit Partitionen anhand von Listenzugehörigkeit definiert sind.

- **HASH-Partitionierung**
  Es erfolgt eine Partitionierung zu einem Spaltenattribut. Zusätzlich muss die Anzahl der Partitionen festgelegt werden. Dementsprechend viele Partitionen werden erstellt und automatisch durch das Datenbankmanagementsystem initialisiert und bestimmt.

- **KEY-Partitionierung**[9]
  Ist fast identisch mit der *HASH*-Partitionierung, mit dem Unterschied, dass nicht ein Hash eines beliebigen Attributs gebildet wird sondern anhand eines Schlüssels (Identifikationsnummer). Es handelt sich um unterschiedliche Algorithmen, die automatisch die Partitionierungswerte errechnen.

Zusätzlich zu diesen Methoden ist es möglich eine Unterpartitionierung durchzuführen, die ab der MySQL Version 5.1 unterstützt wird. Damit ist die weitere Partitionierung von bereits partitionierten Tabellen gemeint. Seit der MySQL Version 5.1 und der heute vorliegenden Version 6.0 kann die Unterpartitionierung bei Partitionen durchgeführt werden, die mit den Methoden *RANGE* und *LIST* erstellt wurden [MySQL August 2009c, Kapitel 17.2.5. Subpartitioning]. Für die Unterpartitionierung stehen ebenfalls die vier Methoden zur Verfügung. Die Funktionen werden in Kapitel 4.3 bespielhaft für MySQL verdeutlicht.

2.4.2. MySQL Datenbank Server

Im Institut für Verkehrssystemtechnik des DLR e.V. wird überwiegend MySQL als Datenbank Server und Verwaltungssystem verwendet. Es handelt sich bei MySQL um das weltweit populärste Open Source Datenbanksystem. Open Source bedeutet, dass der Source Code frei verfügbar ist. Somit wird eine individuelle Weiterentwicklung ermöglicht. Auf Grund dessen wird an dieser Stelle speziell auf MySQL eingegangen.

[9] wird auch Reference-Partitionierung genannt

Entwicklung von MySQL

MySQL wurde von der schwedischen Firma MySQL AB 1994 zum ersten Mal veröffentlicht. Weiterhin baut MySQL auf SQL (Structured Query Language), die geläufigste standardisierte Sprache für Datenbank Handling [MySQL August 2009b, Kapitel 1.4. Was ist MySQL?].
Das Datenbanksystem ist im Vergleich zu der Konkurrenz mit der Entwicklung für Partitionierung im Rückstand. Beispielsweise Oracle führte die *RANGE*-Partitionierung bereits in der Version 8 im Dezember 1997 ein [Oracle Dezember 1997]. Für MySQL ist Partitionierung seit der Version 5.1 möglich, deren erste Veröffentlichung im November 2005 erschien [MySQL August 2009b, Kapitel 17. Partitionierung]. Im Februar 2008 wurde MySQL AB von Sun Microsystems übernommen [MySQL 2008]. Am 20. April 2009 übernahm Oracle Sun Mircrosystems und somit auch MySQL [Microsystems April 2009].

3. Analyse und Ziele der Traffic-Data-Platform (TDP)

3.1. Beschreibung der TDP

Im Institut für Verkehrssystemtechnik des DLR e.V. entstand die Motivation eine Plattform zu entwickeln, in der verteilte Prozesse und Dienste zusammenhängen und -wirken. Diese stehen überwiegend im Zusammenhang mit Verkehrsdaten und deren Verarbeitung und Verwendung. Durch diese Plattform werden die Anwendungen "unter einem Dach" gehalten, sie liegen nicht unabhängig voneinander zerstreut vor und können besser interagieren. Die Dienste und Applikationen werden vereint und es wird die Erstellung von neuen Anwendungen vereinfacht. Dies wird beispielsweise dadurch ermöglicht, dass Entwickler sich nicht um Übertragungsmechanismen und Technologien kümmern müssen. Konvertierungen können in einem einheitlichen Format in der TDP vorgenommen werden. Zudem entfällt die Punkt-zu-Punkt Verbindung zwischen Anbieter und Nutzer. Anwender können direkt über die Plattform auf ein weit gefächertes Angebotsfolio zurückgreifen. Verkehrsdaten können sowohl intern als auch extern verfügbar gemacht werden. Weiterhin können die Daten analysiert, aufbereitet, verarbeitet oder fusioniert werden. Unabhängig von verteilten Systemen und Daten wird in der Plattform der Zugriff und Austausch von Verkehrsdaten erleichtert, und für weitere Entwicklungen und Forschungen werden Ergebnisse und Auswertungen bereitgestellt. Im Anhang B wird der Systemaufbau der TDP gezeigt. Die Plattform besteht aus den folgenden Komponenten:

- Data-Input-Interface und DataImporter
- Prozessierungs-Module
- Service-Module
- Data Output/Service Interface
- Management Konsole
- User-Management (Security)

- Map Server
- Database Management System (DBMS)

Die Plattform verfügt über Schnittstellen für Dienste und Daten und ist aus ein oder mehreren Service- und Prozessierungs-Modulen aufgebaut. In der TDP befindet sich des Weiteren ein Datenverwaltungssystem, ein Map Server und für den Eingabebereich und die Verwaltung eine Konsole. Es existiert eine Komponente für die Sicherheit der Nutzer, also auch Benutzerrechte und -verwaltung. Ein Prozessierungs-Modul kann zum Einen ein Modul für die Verarbeitung von gesendeten Verkehrsdaten sein oder zum Anderen auch ein Modul, das auf die verarbeiteten Daten eines anderen Prozessierungs-Moduls zugreift. Als Beispiel wäre ein Modul zu nennen, das GPS empfangene Daten durch intelligente Algorithmen und Kartenabgleiche den tatsächlichen Positionen zuweist. Denn von GPS empfangene Daten sind nicht immer exakt und liegen eventuell nicht auf der eigentlich gefahrenen Strecke. Verkehrsdaten werden durch Kamera, GPS Daten von FCD-Teilnehmern und Schleifendaten erhalten. Als Service-Modul wäre zum Beispiel ein Dienst zu nennen, der bei einer Anfrage für eine Route die Informationen anhand der aktuellen Verkehrslage ausgibt. Es wird eine Routenführung gegeben, die hohen Verkehrsdichten ausweicht. Dies wird durch zur Laufzeit bekannte prozessierte Verkehrsdaten ermöglicht.

3.2. Vorteil von SOA in der TDP

Der Struktur, beziehungsweise dem Aufbau und der Ziele der TDP, kommt das Systemparadigma einer Serviceorientierten Architektur in vieler Hinsicht zugute. Die modularen Prozessierungsabschnitte und Dienste können autonom voneinander in der Plattform vorliegen und dynamisch zur Laufzeit je nach Anforderung gebunden werden. Einzelne Anwendungsabschnitte und Dienste werden so abstrakt definiert, dass ein hoher Grad an Wiederverwendbarkeit gegeben ist. Die SOA bietet der TDP und deren verteilten Anwendungen und Diensten eine Flexibilität und Unabhängigkeit der verwendeten Technologien. Entwickler können autonom Anwendungen erstellen und müssen lediglich das Schnittstellenkonzept der TDP einhalten. Weiter ermöglicht die SOA bessere Interoperabilität der verschiedenen Module.

3.3. Web Service Entwicklung in der TDP

Die Entwicklung von einem Web Service erfolgt in der TDP mit dem Framework Axis und für die Erzeugung des Programmes (Code Generierung, Kompilierung) Ant, die beide von Apache zur Verfügung gestellt werden. Es wird eine veraltete Version von Axis verwendet, wobei zur Anfertigung dieser Arbeit Axis2 in der

Version 1.5 [Apache 2009b] und Ant in Version 1.7.1 [Apache 2009a] vorliegen. Axis steht für Apache eXtensible Interaction System und kann in C++ und Java eingesetzt werden. Axis ist eine SOAP-Funktionseinheit, die eine automatische Erzeugung der Web Service Schnittstelle (WSDL) unterstützt. Entwickler brauchen sich somit nicht um das Protokoll (SOAP) und die Web Service Schnittstelle (WSDL) zu kümmern, sondern nur um die relevanten Funktionseinheiten. Für die Erzeugung des Programmes wird Ant verwendet. Durch Ant kann mit Hilfe eines *build.xml* File in der entsprechenden Web Applikation das Programm, ein WAR-File (Web Archive), kompiliert werden.

3.4. Optimierungsbedarf in der TDP

Die in der Traffic-Data-Platform übertragenen und verarbeiteten Verkehrsdaten sind beispielsweise GPS-Rohdaten von FCD-Teilnehmern und Schleifen- und Kameradaten aus vielen verschiedenen geographischen Bereichen, die eine beträchtliche Datenmenge produzieren. Diese große Datenmenge muss übertragen, organisiert und gespeichert werden. Daher nimmt die Optimierung datenintensiver Prozesse in der Traffic-Data-Platform eine bedeutende Aufgabe ein. Auf Grund der vorliegenden datenintensiven Prozessen im System wurden für die Optimierung in der TDP drei Bereiche identifiziert, denen eine Verbesserung signifikant zugute kommt. Dabei handelt es sich um folgende Bereiche:

1. Übertragung (auch Kommunikation) zwischen Client und Server (von extern zu intern)
2. Datenspeicherung und -verwaltung (intern)
3. SOA der Plattform und Web Services (intern)

3.4.1. Optimierungsbedarf der datenintensiven Übertragung

Die Übertragungen erfolgen nach dem Simple Object Access Protocol (SOAP). Wie in Kapitel 2.2.4 beschrieben, basiert dieses Protokoll auf XML. Aus der Syntax von XML ergibt sich, dass Daten durch das Tag-System mit einem hohen Zeichenaufkommen beschrieben werden, die Datenbestände werden "aufgebläht" und unübersichtlich. Vor allem bei hohen Datenbeständen nimmt dies ein unkontrolliertes Ausmaß an. Müssen nun zusätzlich große Mengen an Daten in XML Format umgeformt werden, bedeutet das einen sehr großen Aufwand und signifikante Einbußen der Performance. Hier wird die Notwendigkeit deutlich, ein geeigneteres Format für die Übertragung zu finden.

3.4.2. Bedarf an organisierter und strukturierter Datenspeicherung

Bei hohem Datenaufkommen ist es zum Einen wichtig ein geeignetes Datenverwaltungssystem und Speicherformat zu wählen und zum Anderen die Daten effektiv zu speichern und zu verwalten. Ein gutes Datenverwaltungssystem hat nicht zwangsläufig eine effektive Speicherung zur Folge. Je nach Anforderungen gilt es eine geeignete Speicherform zu finden und zu verwenden. Größtenteils unterliegt es dem Administrator das Datenverwaltungssystem gut einzusetzen, es müssen ideale Datenaufteilungen und -modelle konzipiert und eingestellt werden.

3.4.3. Richtige Umsetzung der SOA und Web Service Erzeugung

Damit Funktionalitäten zur Verfügung stehen, die eine SOA auszeichnen[1], gilt es das Systemparadigma richtig einzusetzen. Dafür müssen zuerst angemessene Anforderungen an das System aufgestellt werden. Vor allem ist es wichtig Schnittstellen oder Erfordernisse zu identifizieren, die heterogene Programmiersprachen und Technologien unterstützen. Die Entwicklung von neuen Web Services soll schnell und einfach funktionieren. Hierfür müssen entsprechende Entwürfe vorliegen.

[1] bspw. das dynamische Binden von Services

4. Systemkonzept für die Datenorganisation und -speicherung

4.1. Vorteile durch Partitionierung

Ein sinnvoller Ansatz für die effektive Verwaltung ist die Partitionierung von Datenbanktabellen, vor allem für Datensätze von hoher Kapazität. Partitionierung ermöglicht eine saubere Speicherung und eine Steigerung der Effizienz, indem nur noch Teilblöcke bearbeitet werden müssen, die Informationen zu einer Anforderung enthalten. Partitionierung erfolgt transparent[1], das heißt die Partitionierung bleibt verborgen. Die Anwendungen und Nutzer greifen weiterhin auf nur ein System zu, das physikalisch aber aus mehreren Teilen besteht. Durch die Transparenz wird gewährleistet, dass bei Änderungen oder Erneuerungen von Datensätzen nur die jeweiligen betroffenen Partitionen angepasst werden müssen und nicht das gesamte System.
Intelligente Datenbankmanagementsysteme erkennen bei Abfragen, welche Teiltabellen für die Abfrage benötigt werden, und lassen die Segmente außer Acht, die nicht der Selektion zugehören. Diese Eigenschaft eines Datenbanksystems wird auch als Partition Pruning (Ausschluss von Partitionen) bezeichnet, das in dem folgenden Kapitel 4.1.1 erläutert wird. Hieraus resultiert ein bedeutender Performancevorteil. Denn für umfangreiche Abfragen können große Datenbereiche, die nicht betroffen sind, ignoriert werden.

4.1.1. Partition Pruning

Partition Pruning meint den leistungsfähigen Ausschluss von aufgeteilten Datenbanktabellen bei einer Abfrage. Anfragen werden vor der Durchführung analysiert und nur betroffene Partitionen einbezogen. Um einen Ausschluss von Partitionen zu unterstützen und somit eine Beschleunigung abzuarbeitender Prozesse zu erzielen,

[1] mit Transparenz ist in der Informatik die Soft- oder Hardware gemeint, die für Anwender unbemerkt bleibt

empfiehlt es sich in Abfragen der Partitionen *WHERE*-Klauseln zu verwenden. Zudem muss ein Wertebereich mit *BETWEEN* oder durch die mathematischen größer, kleiner und gleich Zeichen ($<$, $>$, $=$) gebildet werden. Im Bezug auf einen Wertebereich können dazugehörige, beziehungsweise Partitionen mit entsprechendem Inhalt zugewiesen werden.

4.2. Voraussetzungen für Partitionierung

In MySQL ist Partitionierung und Unterpartitionierung seit der Version 5.1 möglich. Ab der MySQL Version 5.1.6 wurde die Funktionalität von Partition Pruning implementiert [MySQL August 2009a, Kapitel 18.4. Partition Pruning]. Daher wird empfohlen minimal die Version 5.1.6 von MySQL zu verwenden, um die Perfomance Vorteile von Partitionierung voll auszunutzen.

4.3. Partitionierung von Datenbanken am Beispiel von MySQL

Wie in Kapitel 2.4.1 genannt, stehen für die Partitionierung von Datenbanktabellen vier Methoden zur Verfügung. Anhand eines Beispiels für eine Funktion, die *RANGE*-Partitionierung, werden diese nun in Bezug auf MySQL veranschaulicht. Eine *RANGE*-Partition kann, wie in Abbildung 4.1 dargestellt, zusammen mit einer *CREATE TABLE* Klausel umgesetzt werden. Als Parameter wird der

```
1  CREATE TABLE bsp
2  (
3    ....
4    a_date DATE NOT NULL,
5    ....
6  )
7  PARTITION By RANGE (year(a_date))
8  (
9    PARTITION part1 VALUES LESS THAN(2005),
10   PARTITION part2 VALUES LESS THAN(2007)
11 );
```

Abbildung 4.1.: Partitionierung bei MySQL am Beispiel *RANGE*

Methode die Funktion *year()* gesetzt, die anhand eines Datums das Jahr als Integer zurückgibt. Die Partitionen werden mit den Werten kleiner als (*VALUES LESS THAN*) eingeteilt. Somit ist klar definiert, in welche Bereiche zugeordnet wird. Wird versucht Werte hinzuzufügen, die größer als der Höchstwert sind (in dem Beispiel größer als 2007), werden diese außer Acht gelassen und nicht in der Tabelle und deren Partitionen aufgenommen. *RANGE*- und *LIST*-Partitionierungen können in weitere Teilpartitionen zerlegt werden. Für diese Teilpartitionierung stehen

in MySQL nur *HASH*- und *KEY*-Partitionierung zur Verfügung. In Abbildung 4.2 wird die Unterpartitionierung einer *RANGE*-Partition mit *HASH*-Partitionierung veranschaulicht.

```
1  CREATE TABLE bsp
2  (
3    ....
4    a_date DATE NOT NULL,
5    ....
6  )
7  PARTITION By RANGE (year(a_date))
8   SUBPARTITION BY HASH(month(a_date))
9   SUBPARTITIONS 4
10 (
11   PARTITION part1 VALUES LESS THAN(2005),
12   PARTITION part2 VALUES LESS THAN(2007)
13 );
```

Abbildung 4.2.: Unterpartitionierung bei MySQL

4.4. Vorarbeit für Partitionierung

Auf Basis der Vorarbeit wird das Systemkonzept erarbeitet um die Datenorganisation und -speicherung in der TDP zu verbessern. In Kapitel 4.1 wurden die Vorteile erläutert, die eine Partitionierung von Datentabellen auszeichnen und die vor allem bei einer Umsetzung für diese Arbeit als signifikant identifiziert wurden. Aus diesen Vorteilen entstand die Motivation, Partitionierung für die TDP und allgemein für das Institut für Verkehrssystemtechnik des DLR e.V. und deren Datenverwaltungssysteme einzusetzen. Wie in Kapitel 2.4.2 beschrieben, wird MySQL als Datenverwaltungssystem bevorzugt. Deshalb wurden in Kapitel 4.3 allgemeingültig die Grundlagen für eine Partitionierung mit MySQL dargelegt. Nachdem der aktuelle Stand und die Verwendung des Datenverwaltungssystems analysiert wird (Kapitel 4.5), kann auf Basis der Ergebnisse ein Entwurf basierend auf der aktuellen Datenorganisation entworfen werden (Kapitel 4.6), mit dem die Partitionierung der Datenbanktabellen für das Institut umgesetzt werden kann (Kapitel 4.6.1). Diese Umsetzung erfolgt beispielhaft mit einer verwendeten Datenbanktabelle des Instituts. Anhand einer zweiten Tabelle wird die Effizienz und Performance getestet, indem Abfragen, Beobachtungen und Kontrollen durchgeführt und verglichen werden. Diese Tabelle verfügt über den gleichen Datenbestand, wird aber nicht partitioniert.

4.5. Analyse der Datenorganisation

Wie schon in den vorherigen Kapiteln beschrieben, wird für die Verwaltung von Verkehrsdaten ein MySQL Datenverwaltungssystem verwendet. Für die Analyse

der Datenorganisation werden vor allem Konzepte und Aufbau der Datenhaltung betrachtet. Die gespeicherten Informationen beinhalten unter Anderem Identifikationsnummer, Zeit, Geschwindigkeiten und Positionsattribute. Die Datenlieferanten sind Kooperationspartner, wie der öffentliche Nahverkehr oder Taxiverbände aus verschiedenen Städten. Um die großen Datenmengen skalieren zu können und die Auslastungen einer Tabelle zu verringern, wird für jedes Jahr einer Stadt eine Datentabelle angelegt. Diese Aufteilung wird manuell durchgeführt. Somit wird eine Stadt schon auf viele Tabellen aufgeteilt und die Anzahl der Tabelle nimmt jährlich zu. Es handelt sich bei dieser Aufteilung um eine sowohl logische als auch physikalische Partitionierung von Datenbanktabellen, die manuell angelegt wird. Mit der jährlich steigenden Anzahl der Tabellen wird die Übersichtlichkeit zunehmend erschwert. Zudem ist es nicht möglich eine jahresübergreifende Abfrage einer Tabelle durchzuführen, ohne mindestens zwei Verbindungen zu je einer Tabelle mit einem Jahr aufzubauen. Weiterhin ist auch die manuelle Aufteilung aufwendig. Auf Grund dieser Tatsachen wird bei dem Anwender, beziehungsweise Administrator das Interesse geweckt eine rein physikalische und automatische und technische Partitionierung umzusetzen. Für die MySQL Datenverwaltung wird My Indexed Sequential Access Method (MyISAM) als Tabellentyp verwendet. Bei diesem Typ werden drei Dateien für eine Tabelle erzeugt:

1. Eine Formatdefinitions-Datei (.frm Dateiendung). Sie enthält Definitionen und Beschreibungen der Tabelleneigenschaften.

2. Eine Daten-Datei (.myd Dateiendung (My Data)). Sie enthält die Daten der Tabelle.

3. Eine Index-Datei (.myi Dateiendung (My Index)). Sie enthält eine Übersicht der Tabelle und fungiert als Verzeichnis.

4.6. Detaillierter Entwurf der Partitionierung auf Basis der bestehenden Datenorganisation

Wie in Kapitel 2.4.1 beschrieben stehen für eine Partitionierung vier Methoden zur Verfügung. Damit eine Aufteilung für die Verkehrsdatenverwaltung umgesetzt werden kann, gilt es zuerst zu untersuchen und zu überprüfen, welche Art erforderlich ist und welche Methoden in Frage kommen. Als Ergebnis der Analyse der bestehenden Datenorganisation im Kapitel 4.5 kann festgehalten werden, dass bereits manuell eine logische und physikalische Aufteilung der Daten nach Jahren und Städten vollzogen wird. Durch die jährlich äußerst hohen Datenaufkommen und die Beschaffenheit der Daten zeigt sich eine Hauptaufteilung der Daten einer Stadt nach Jahren als vorteilhaft. Mit der Beschaffenheit von Daten ist der Gehalt an Informationen eines Speichersatzes gemeint. Jeder Datensatz beinhaltet immer einen Zeitstempel.

Daraus folgt die Anforderung, auch die funktionale Partitionierung nach Jahren aufzuteilen, also nach einem Wert. Diese Anforderung gewährleisten die Methoden *RANGE* und *LIST*-Partitionierung, denn bei diesen ist es möglich ein Jahr als Partitionierungswert anzugeben. Bei *RANGE*-Partitionierung geschieht die Verteilung nach Wertebereichen und bei *LIST*-Partitionierung nach Wertelisten. Diese Methoden kommen deshalb für eine Wertepartitionierung in Frage. Des Weiteren kann eine *HASH*-Partitionierung als Unterpartitionierung angewendet werden, um zusätzlich zu der Jahresaufteilung noch die Monate aufzuteilen, zum Beispiel vierteljährlich. Die *KEY*-Partitionierung ist für ein Aufteilen nach Schlüsseln vorgesehen, wie Fremdschlüssel oder Primärschlüssel, was für diese Anwendung nicht von Bedeutung ist. Es kann demzufolge für weitere Untersuchungen außen vor gelassen werden. Die Secure Query Language stellt Funktionen zur Verfügung um von einem Zeitstempel nur bestimmte Zeiteinheiten wie Jahr oder Monat zu erhalten. Diese sind notwendig um die Datensätze nach einen Jahr zu partitionieren, es wird die Funktion *year()* angewendet. Die funktionale Partitionierung soll auszeichnen, dass für jede Stadt nur eine Tabelle benötigt wird, es verbessert die Übersichtlichkeit, und Abfragen können Jahre übergreifend durchgeführt werden. Die Performance von Abfragen soll trotz des Zusammenschlusses von Daten durch die jahrweise physikalische Trennung und durch die Verwendung des Features von Partition Pruning optimiert sein.

4.6.1. Verwendung und Durchführung der Partitionierung

Für die Durchführung wurde eine Stadt ausgewählt, in der ein hohes Datenaufkommen vorliegt. Und zwar *positions_ copenhagen*, in der FCD Positionsdaten von Taxis gespeichert werden. Anhand dieser Tabelle wird beispielhaft Performance und Nutzenvorteil der Partitionierung analysiert. Es wird bei MySQL ein Test-Schema angelegt, in dem für diese Stadt vier verschiedene Tabellen mit unterschiedlichen Partitionierungstypen angelegt werden. Damit diese Tabellen möglichst effizient erstellt werden, gilt es zuerst die Eigenschaften der Partitionierungs-Methoden genauer zu begutachten. Hash Partitionierung nimmt nach folgender Systematik die Einteilungen vor:

- Das Ergebnis aus *MOD(key, Partitionen)* ergibt die Partition, in der der Inhalt gespeichert wird, zum Beispiel bei vier Partitionen und dem Datum vom 01.März 2009 *year(2009-03-01) -> MOD(2009,4) = 1 ->* Speicherung in der Partition 1.
- *RANGE*-Partitionierung legt die Daten in die Partition, in welcher die Wertebereiche mit den Datenwerten übereinstimmen.
- Bei *LIST*-Partitionierung werden die Daten je nach zutreffendem Wert in einen angegebenen Listenbereich gesetzt.

- Seit der Version 5.1.8 müssen auch Subpartitionen eindeutige Namen haben. Vorher durften Unterpartitionen verschiedener Partitionen gleiche Namen besitzen [Vgl.: MySQL August 2009a, Kapitel 18.2.5 Subpartitioning].

Folgende vier Tabellen mit unterschiedlichen Partitionierungsschemata werden angelegt:

1. *RANGE*-Partitionierung nach Jahr
2. *RANGE*-Partitionierung nach Jahr und *HASH*-Unterpartitionierung nach Monat
3. *LIST*-Partitionierung nach Jahr
4. *LIST*-Partitionierung nach Jahr und *HASH*-Unterpartitionierung nach Monat

Skripte, die bei Erstellung der partitionierten Tabellen angewendet werden, sind auf der mitgelieferten CD unter dem Verzeichnis Datenorganisation zu finden. In dem Test-Schema werden zusätzlich Kopien von den Tabellen für die Jahre 2008 und 2009 angelegt(*positions_copenhagen_2008* und *positions_copenhagen*), um praxisorientierte Versuche durchzuführen, ohne jedoch die in der Praxis verwendeten Tabellen modifizieren und belasten zu müssen. Die partitionierten Tabellen werden mit dem Inhalt dieser beiden Tabellen gefüllt. Zusätzlich wird eine weitere Tabelle angelegt, die mit dem gleichem Inhalt gefüllt, aber nicht partitioniert wird. Anhand dieser Tabelle kann der Performancevorteil bei gleichem Datengehalt verglichen werden, den eine partitionierte Tabelle eventuell ausmacht.

4.6.2. Annahme der Performancevorteile

Bevor die genauen Testdurchführungen beschrieben und vollzogen werden, wird eine These aufgestellt, welche Ergebnisse erwartet werden. Hierbei handelt es sich um eine Annahme, welche Performancevorteile und -nachteile bei den unterschiedlichen Tabellentypen im Vergleich vermutet werden. Dabei wird auch angenommen, dass Testabfragen so durchgeführt werden, dass die Vorteile einer Partitionierung deutlichen werden können[2]. Folgende Annahmen werden aufgestellt:

1. Bei den Tabellen, die beide Jahre als Inhalt haben, wird die Tabelle mit dem Partitionierungstyp 1 (Erst *RANGE* Partitionierung nach Jahr, dann *HASH* nach Monat) am besten abschneiden. Diese Vermutung begründet sich darauf, dass bei Zeitabfragen am schnellsten zuerst jahrweise und danach monatsweise Tabellenteile ausgeschlossen werden können, die nicht betroffen sind.

[2]zum Beispiel mit Ausnutzen von Partition Pruning

2. Jahresübergreifende Fragen können von den manuell geteilten Tabellen zwar nicht ausgeführt werden, aber die Tabellen können nacheinander so abgefragt werden, dass die gleiche Datenmenge erhalten wird wie bei einer jahresübergreifenden Abfrage. Es wird angenommen, dass alle anderen Tabellen, in denen die Daten zusammenhängend vorliegen und partitioniert sind, besserer Performance unterliegen.

3. Es wird vermutet, dass die Tabelle, in der die Daten für zwei Jahre gespeichert sind, aber keine Partitionierung vorliegt, allgemein am schlechtesten abschneidet, da immer eine größere Datenmenge komplett berücksichtigt werden muss.

4.7. Performance Test und Auswertung

Für den Performance Test ist es relevant, welche Zeit das MySQL Datenverwaltungssystem benötigt eine Abfrage abzuarbeiten und Ergebnisse anzuzeigen. Anhand dieser Zeit kann verglichen werden, welcher Tabellentyp am schnellsten arbeitet und mit einer Abfrage am besten umgehen kann. Für solche zeitkritischen Überprüfungen ist es von höchster Wichtigkeit die Testumgebung zu beachten, denn schon kleine Unregelmäßigkeiten, wie zum Beispiel paralleler Zugriff eines weiteren Nutzers und Durchführungen von speicherintensiven Programmen, beeinträchtigen die Ergebnisse signifikant und können sie somit verfälschen. Damit für diese Testdurchführungen verfälschte Ergebnisse erkannt werden und dementsprechend eine Verfälschung unwahrscheinlicher ist, werden Abfragen in bestimmten Abständen dreimal durchgeführt. Zudem wird eine Testumgebung gewählt, in der sehr viel Arbeitsspeicher zur Verfügung steht und das System somit relativ konstante Leistung für Anwendungen erbringt. Die genaue Testumgebung wird im Anhang C.1 angegeben. Die Testdurchführungen werden mit vier verschiedenen Datenbankabfragungen durchgeführt. Bei den Abfragen wurde vor allem darauf geachtet, dass die Vorteile einer Partitionierung deutlich werden können. Auf Grund dessen werden zeitliche Einteilungen oder auch Bereiche speziell ausgewählt. Desweiteren wurde eine Abfrage ausgewählt und verwendet, die in der Praxis sehr häufig benötigt wird, sodass praxisnah und -relevant die Tests durchgeführt werden. Für diese Unternehmungen wurde das Programm MySQL Query Browser verwendet[3], das von MySQL als GUI Tool zur Verfügung gestellt wird. Schließlich ist das Programm benutzerfreundlich aufgebaut und zeigt bei Durchführungen wichtige Systemausgaben auf. Bei ergebnisliefernden Befehlen wird von dem Programm die benötigte Zeit in folgender Form dargelegt:

"X row(s) fetched in Y s (Z s)"

- "X" ist die Anzahl der Zeilen.

[3]siehe: C.2, ein Abbild des Programms

- "Y" ist Zeit, die benötigt wird, um das Ergebnis vom Server zum Nutzer zu übertragen.
- "Z" ist die Zeit, die der MySQL-Server braucht um den Befehl durchzuführen (analysieren des Befehls, abrufen und senden der Daten).

Für weitere Untersuchungen ist nur die Zeit "Z" von Bedeutung. Diese steht für die Dauer um den Befehl durchzuführen und die zutreffenden Daten zu erhalten. "Y" steht für die Zeit, die gebraucht wird um dem Benutzer das Ergebnis zu überliefern und anzuzeigen. Eine Addition der beiden Zeiten würde die Zeit ergeben, die von der Durchführung bis hin zum Anzeigen erforderlich ist.
Die Ergebnisse dieses Performance Tests können im Anhang eingesehen werden.

4.7.1. Auswertung der Ergebnisse

Entgegen der Erwartungen und Annahmen ergibt sich aus dem Performance Test kein Vorteil der partitionierten Tabellen. Im Vergleich zu einer nicht partitionierten Tabelle, die über den gleichen Datenbestand verfügt, bestehen sie sogar fast identisch. Auf Grund dessen entsteht die Motivation die Ursachen für diese Ergebnisse aufzuzeigen, und eventuell Verbesserungen durchzuführen. Folgende mögliche Begründungen werden aufgestellt:

- MySQL speichert die Ergebnisse in einem Art Cache. Dadurch zeigen sich sowohl partitionierte als auch nicht partitionierte in ähnlicher Performance. Denn auch bei nicht partitionierten Tabellen kann dank des Cache auf eine Art Register und Verweise zugegriffen werden.
- Die Partitionierung mit einer Funktion als Partitionierungsschlüssel ist zu aufwendig und verschlechtert die Performance.
- Das MySQL System wurde noch nicht auf die Partitionierung zugeschnitten. Damit ist die passende und optimimale Verwendung von MyISAM und Systemvariabeln im Zusammenhang mit Partitionierung gemeint.

4.7.2. Ursachen der Ergebnisse

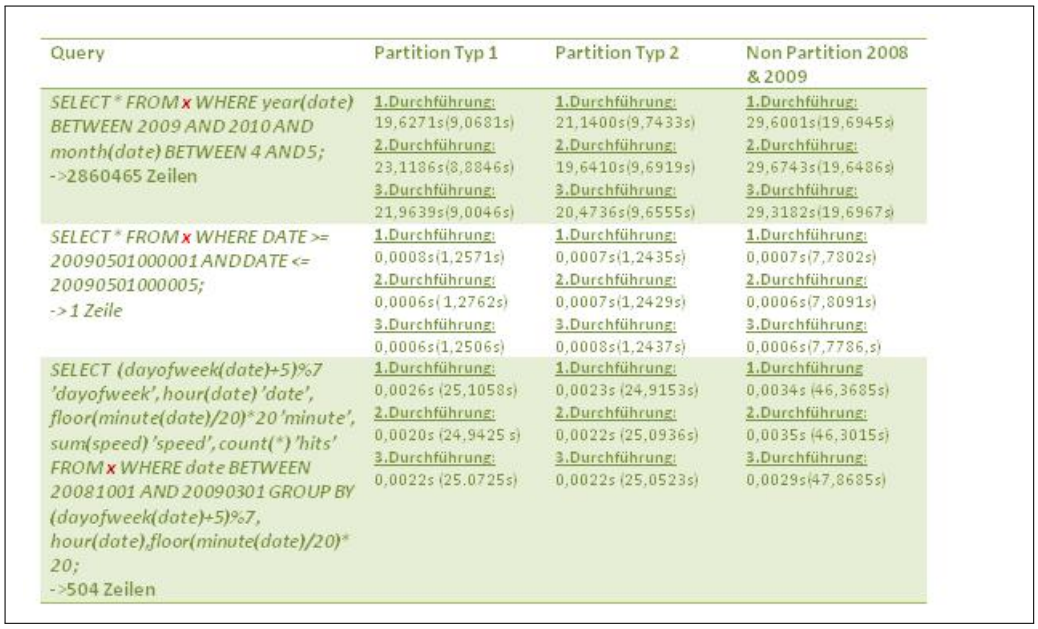
Nach genauen Untersuchungen wird festgestellt, dass die Testumgebung keinen Cache in der MySQL Anwendung verwendet. Mit dem Programm MySQL Administrator, das zusammen mit MySQL Query Browser als GUI Tool von MySQL angeboten wird, kann unter dem Menüpunkt Startvariablen und dem Untermenüpunkt Performance die Einstellungen für Cache Size nachvollzogen werden. Diese Variabel ist auf null gesetzt und wird somit nicht angewendet. Ebenso ist es möglich die von MySQL verwendeten Variablen, zum Beispiel im MySQL Query Browser, durch

den Befehl *SHOW VARIABLES* anzeigen zu lassen.
Die Annahme, dass die Partitionierung durch die Handhabung einer Funktion im Partitionierungsschlüssel deutlich in der Performance geschwächt wird und dadurch die unerwarteten Ergebnisse entstehen, wird als unwahrscheinlich eingestuft. Es ist unrealistisch, dass die Verwendung von Funktionen die Partitionierung verschlechtert. Denn das Ergebniss zeigte fast identische Performance von partitionierten und nicht partitionierten Tabellen. Dies lässt eher darauf schließen, dass die Vorteile der Partitionierung nicht genutzt werden oder Tabelleneigenschaften nicht richtig gewählt sind.
Wie in Kapitel 4.5 beschrieben, werden bei MyISAM für eine Tabelle drei Dateien angelegt. Bei der Partitionierung wird beobachtet, dass für jede Tabelle eine Formatdefinition-Datei angelegt wird und für jede Partition eine Daten- und Index-Datei. Dabei wird erkannt, dass Index-Dateien im Allgemeinen, also auch bei nicht partitionierten Tabellen, sehr groß sind und viel Speicher verwenden. Bei den Testdurchführungen belief sich die Größe der Datendatei für das Jahr 2008 auf über 1,6 Gigabyte und die Index-Datei etwas weniger als die Hälfte der Daten-Datei. Dieses Verhältnis wird bei jeder Daten- und Index-Datei gesehen. Aus diesem Grund werden die Indexreferenzen der Tabellen überprüft. Hierdurch wird ein *dateindex* der Tabellen festgestellt, der für den Datentyp *TIMESTAMP* gesetzt wird. In den Indexdateien ist somit eine Referenz für die Zeiten erstellt. Dadurch wird viel Speicher benötigt, aber Abfragen, die Zeiten beinhalten, können schneller ausgeführt werden. Diese Referenz ist auch die Ursache für die fast identische Performance der partitionierten und nicht partitionierten Tabellen. Des Weiteren wurde bei den Recherchen über die Ursachen der Performance Testergebnisse festgestellt, dass der MySQL Datentyp *TIMESTAMP* nicht Partition Pruning unterstützt, sondern die Datentypen *DATETIME* oder *DATE* [Vgl.: MySQL August 2009a, Kapitel 18.4 Partition Pruning]. Stattdessen empfiehlt es sich *DATETIME* oder *DATE* zu verwenden. Auf Basis dieser Auswertungen wird der Performance Test erneut durchgeführt und im folgenden Kapitel 4.7.3 beschrieben. Hierfür wird zum Einen der *dateindex* entfernt und zum Anderen *DATETIME* statt *TIMESTAMP* als Datentyp für die Zeit verwendet. Dabei gilt zu beachten dass *TIMESTAMP* 4 Byte groß ist und *DATETIME* 8 Byte. Obwohl *DATETIME* doppelt so groß ist, wird bei diesen Durchführungen nur ein Größenzuwachs von ungefähr 20 Prozent bei den Datendateien beobachtet.

4.7.3. Erneute Durchführung des Tests auf Basis der Auswertungen

Die erneute Durchführung eines Performance Tests wird anhand von zwei partitionierten Tabellen im Vergleich zu einer nicht partitionierten Tabelle verdeutlicht. Dafür wurden den Tabellen der *dateindex* entfernt und für die Zeit wird statt dem

Datentyp *TIMESTAMP DATETIME* verwendet. Eine partitionierte Tabelle verfügt zusätzlich noch über Unterpartitionen (Typ 2). In Abbildung 4.3 werden die Ergebnisse der modifizierten Durchführung gezeigt. Bei diesem Test wird nun der



Query	Partition Typ 1	Partition Typ 2	Non Partition 2008 & 2009
SELECT * FROM x WHERE year(date) BETWEEN 2009 AND 2010 AND month(date) BETWEEN 4 AND5; ->2860465 Zeilen	1.Durchführung: 19,6271s(9,0681s) 2.Durchführung: 23,1186s(8,8846s) 3.Durchführung: 21,9639s(9,0046s)	1.Durchführung: 21,1400s(9,7433s) 2.Durchführung: 19,6410s(9,6919s) 3.Durchführung: 20,4736s(9,6555s)	1.Durchführug: 29,6001s(19,6945s) 2.Durchführug: 29,6743s(19,6486s) 3.Durchführug: 29,3182s(19,6967s)
SELECT * FROM x WHERE DATE >= 20090501000001 AND DATE <= 20090501000005; ->1 Zeile	1.Durchführung: 0,0008s(1,2571s) 2.Durchführung: 0,0006s(1,2762s) 3.Durchführung: 0,0006s(1,2506s)	1.Durchführung: 0,0007s(1,2435s) 2.Durchführung: 0,0007s(1,2429s) 3.Durchführung: 0,0008s(1,2437s)	1.Durchführung: 0,0007s(7,7802s) 2.Durchführung: 0,0006s(7,8091s) 3.Durchführung: 0,0006s(7,7786,s)
SELECT (dayofweek(date)+5)%7 'dayofweek', hour(date) 'date', floor(minute(date)/20)*20 'minute', sum(speed) 'speed', count(*) 'hits' FROM x WHERE date BETWEEN 20081001 AND 20090301 GROUP BY (dayofweek(date)+5)%7, hour(date),floor(minute(date)/20)* 20; ->504 Zeilen	1.Durchführung: 0,0026s (25,1058s) 2.Durchführung: 0,0020s (24,9425 s) 3.Durchführung: 0,0022s (25.0725s)	1.Durchführung: 0,0023s (24,9153s) 2.Durchführung: 0,0022s (25,0936s) 3.Durchführung: 0,0022s (25,0523s)	1.Durchführung 0,0034s (46,3685s) 2.Durchführung: 0,0035s (46,3015s) 3.Durchführung: 0,0029s(47,8685s)

Abbildung 4.3.: Modifizierte Test Durchführung

Vorteil von Partitionierung signifikant deutlich. Die Effizienz steigt zum Teil um 200 % bei Befehlen, indem sie doppelt so schnell abgearbeitet werden können. Anhand des zweiten Befehls dieser Anwendung zeigt sich sogar eine Steigerung um 600 %, die Ausführung geschieht sechsmal schneller. Es ist aber kein Vorzug bei Unterpartitionierung zu erkennen, die Partition Typen 1 und 2 zeigen ungefähr identische Performance.

4.7.4. Bewertung des Vorteils von partitionierten Tabellen

Nach einer Anpassung der Durchführung konnte nun der Nutzen einer Partitionierung verdeutlicht werden. Aber die Partitionierung bringt nicht nur eine bedeutende Performance Steigerung mit sich, sondern die Speicherauslastung wird reduziert. Durch eine Partitionierung wird die Indexerstellung überflüssig und somit die hohe Datenmenge an Eintragungen in der Index-Datei. Weiterhin sind nun jahresübergreifende Abfragen möglich. Zudem wird die Datenhaltung für Anwender weitaus übersichtlicher, denn alle Jahre einer Stadt können nun in nur einer partitionierten Tabelle gehalten werden. Ohne Partitionierung erfolgte bei Abfrage eine Berücksichtigung aller Datensätze der Tabelle. Das hat zur Folge, dass der komplette Datenbestand überprüft werden muss, und somit der Arbeitsspeicher unnötig be-

lastet wird. Es werden Daten überprüft, die eigentlich schon vorab außer Betracht gelassen werden können.

4.8. Empfehlungen für die Anwendung und Umsetzung in der Praxis

Die Entwürfe und Durchführung von Konzepten zur Verbesserung der Datenorganisation wurden wesentlich praxisorientiert vollzogen. Dabei wird immer ein Bezug und eine Betrachtung für die Anwendung und Verwendung hergestellt. Vor Allem die Vorteile in Kapitel 4.7.4 werden für die Realisierung der Entwürfe im Institut dargelegt. Auf Grund dessen wird dem Institut empfohlen, die Datenorganisation folgendermaßen anzupassen:

1. Statt einer manuellen Aufteilung der Tabellen nach Jahren, sollte die Partitionierung verwendet werden. Für eine Unterpartitionierung wurde kein Vorteil festgestellt, daher empfiehlt es sich nur eine RANGE-Partitionierung nach Jahren einzusetzen. Denn für jede Partition und Unterpartition wird eine Daten- und Index-Datei angelegt. Für den Anwender der Tabelle scheint dieses zwar verborgen, aber für den Administrator wird die Verwaltung mit Unterpatitionen dadurch zunehmend erschwert und unübersichtlich.

2. Um die Stärke der Partitionierung voll auszunutzen wird empfohlen den *dateindex* zu entfernen und statt den Datentypen *TIMESTAMP DATETIME* oder *DATE* zu verwenden.

4.8.1. Empfohlene Handhabung und Verwaltung der Datenbanken

Bei Umsetzung und Verwendung von partitionierten Tabellen sollten Abfragen möglichst mit *WHERE* und *BETWEEN*-Klausen durchgeführt werden oder mit den mathematischen größer, kleiner und gleich Zeichen ($<$, $>$, $=$) um Partition Pruning zu unterstützen. Es sollten genug Partitionsbereiche angelegt werden und vorsichtshalber auch eine Partition mit *MAX_ VALUE* (maximaler Wert). Alle Werte die größer als die Wertebereiche sind können dadurch in der Partition mit *MAX_ VALUE* gespeichert werden. Dabei sollte aber beachtet werden, dass in der Partition mit *MAX_ VALUE* nicht Daten mit 2 unterschiedlichen Jahren enthalten sind. Die Vorteile der Partitionierung werden dadurch nicht ausgenutzt. Stattdessen empfiehlt es sich die Tabelle anzupassen und eine neue Partition hinzuzufügen.

5. Systemkonzept für die optimierte datenintensive Übertragung

5.1. Analyse der bestehenden Übertragungsmechanismen

Eine Analyse der bestehenden Übertragungsmechanismen benötigt eine geeignete Technik, um den Nachrichteninhalt des verwendeten Übertragungsprotokolls SOAP überprüfen und beobachten zu können. Dafür können Monitor-Systemeinheiten verwendet werden, die den Inhalt der SOAP Nachricht verfolgen, indem sie an die Verbindung zwischen Client und Server andocken. Sowohl bei Axis2 als auch Axis wird im Binary Source Code von Apache ein Programm für die Überwachung der SOAP Übertragung mitgeliefert. Die in Kapitel 3.3 beschriebene aktuelle Version von Axis2 enthält in der Standard Binary Distribution eine *soapmonitor-1.5.jar* Datei, das Programm SOAP Monitor [1]. In Axis und der Version 1.4, die vom Institut des DLR für das Projekt TDP während dieser Arbeit noch verwendet wurde, liegt in der Datei *axis.jar* im Package *org.apache.axis.monitor* ein SOAP Monitor und im Package *org.apache.axis.utils* ein TCP Monitor, der zur Überwachung der TCP-Verbindung (Transmission Control Protocol) verwendet wird. Der TCP Monitor wird für diese Arbeit und Testzwecke für die Überprüfung der SOAP-Übertragungen angewandt. Im Kapitel 5.1.2 wird die genaue Funktionsweise und Durchführung dieses Monitors beschrieben. Was eine JAR-Datei ist und wie sie ausgeführt werden kann, wird im folgenden Kapitel 5.1.1 erläutert.

5.1.1. Java Archive (JAR) Datei

JAR-Dateien sind Java Archive Dateien und können kompilierten Java-Quellcode enthalten und somit als Programm fungieren. Um ein solches Programm ausführen zu können, wird folgend ein Beispiel genannt, das sowohl für Linux als auch

[1] Software Download: Apache Axis2, http://ws.apache.org/axis2/download/1_5/doswnload.cgi, Letzter Zugriff 25.Juni 2009

Windows Betriebssysteme gültig ist. Am Beispiel der *example.jar* Datei, die ausführbar(executeable) ist:

1. Es wird davon ausgegangen, dass Java korrekt installiert ist und in der Eingabekonsole der Pfad, in der sich die Java Archive Datei befindet, ausgewählt wurde.

2. Durch den Befehl *java -jar example.jar* kann nun das Programm ausgeführt werden.

3. Soll nur eine bestimmte, ausführbare Klasse in der JAR ausgeführt werden: *java -cp example.jar org.example.BestimmtesProgramm*

Für detailreichere Informationen, zum Beispiel wie eine JAR ausführbar gemacht werden kann, wird empfohlen diese Beschreibungen in der Dokumentation über Java von Sun zu entnehmen [2].

5.1.2. Anwendung und Funktionsweise des TCP Monitors

Nachdem das Programm des TCP Monitors, wie in Kapitel 5.1.1 beschrieben, ausgeführt wurde, erscheint ein Programmfenster, in dem wichtige Einstellungen getroffen werden müssen.

- *Listen Port:* Hier ist der Port einzugeben, an dem der Monitor "lauscht". Für diese Versuchszwecke wurde der Port 9091 gewählt.

- *Listener/Proxy:* Der TCP Monitor kann entweder als Listener Adressen abhorchen oder einen Proxy untersuchen. Für dieses Projekt wird der Listener ausgewählt. Der TCP Monitor soll für diese Testzwecke die Übertragung zwischen Nutzer und Server beobachten.

- *Target Hostname:* Zieladresse (als IP), an die der lokale Personal Computer, auf dem der TCP Monitor aufgerufen wurde, sendet. Für diese Testzwecke wird der lokale Personal Computer sowohl als Sender als auch Empfänger verwendet, sodass die lokale Adresse als Zieladresse einzugeben ist (127.0.0.1).

- *Target Port:* Port der Zieladresse, auf dem der Server anzusprechen ist (Standard 8080).

Des Weiteren bietet der TCP Monitor die Möglichkeit eine schlechte Verbindung zu simulieren (*Simulate Slow Connection*), und somit können für derartige Fälle entsprechende Testdurchführungen vollzogen werden. Nachdem die Einstellungen

[2] zum Beispiel siehe: [Sun 2002]

hinzugefügt wurden, erscheint ein weiteres Fenster, in dem die Übertragungen beobachtet werden können. Als Format lässt sich XML angeben, sodass die übertragenden Zeichen sinnvoll und übersichtlich angezeigt werden. Im Anhang A.1 wird das Programm TCP Monitor mit den beschriebenen Programm-Fenstern veranschaulicht. Der TCP Monitor kann als Listener zwischen Client und Server dienen. Hierfür müssen entsprechende Einstellungen beim Monitor und beim Sender (Ort, an dem das Programm ausgeführt wird) getroffen werden. Eine wichtige Einstellung ist der Port, an dem der Monitor "lauscht". Dieser Port muss für die Testzwecke beim Client geändert werden. Üblicherweise sendet der Client über Port 8080. Nachdem ein Port beim Client festgelegt wurde (für diese Anwendung Port 9091) und der TCP Monitor an diesem "lauscht", können die gesendeten Daten überprüft werden. Dies geschieht nach der in Abbildung 5.1 gezeigten Funktionsweise des Monitors. Die überprüften Daten werden vom Monitor an den üblichen Port 8080 des Empfängers (Server) weitergeleitet. Antworten des Servers gelangen auch erst für die Überprüfung zum Monitor und werden dann zum Port 9091 des Empfängers (Client) weitergeleitet.

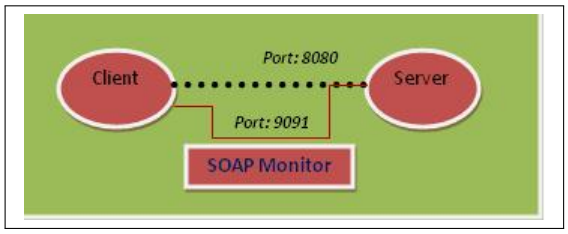


Abbildung 5.1.: Funktionsweise eines SOAP Monitors

5.2. Beschreibung und Analyse der verwendeten Web Services

Um die Beobachtungen und Verbesserungen hinsichtlich der Optimierung von datenintensiver Übertragungen durchzuführen, bietet es sich an, einen Service auszuwählen, der bereits entwickelt wurde und bei dem Übertragungen mit hohen Datenmengen stattfinden. Die Ergebnisse der Beobachtungen werden daraufhin analysiert. Anhand dieser Ausarbeitung können Optimierungsentscheidungen gebildet werden. Es wird für die Untersuchungen das Projekt Smarttruck des Instituts gewählt, das zwei Web Services für die TDP beinhaltet (DLRWebservicesFCD und DLRWebservicesRawFCD). Diese Web Services bieten verschiedene Funktionen an, um zum Beispiel Roh-FCD-Daten oder aggregierte FCD-Daten zu übertragen und zu empfangen. Im Folgenden werden Anwendungsbeispiele für die Web Services genannt:

DLRWebservicesRawFCD:

- Der Server fordert aktuelle Roh-FCD-Daten von einem Client an, worauf der Client die Daten zuschickt.
- Ein Client schickt eine Sammlung von Roh-FCD-Daten an den Server.

DLRWebservicesFCD:

- Ein Client fordert historische FCD-Daten über einen Monat oder aktuelle Daten an, die daraufhin vom Server übertragen werden. Es handelt sich um Daten, die bereits aggregiert wurden.

Um Optimierungsmöglichkeiten und - bedarf festzustellen, werden die Übertragungen von beiden Web Services beobachtet und analysiert.

5.2.1. Problemanalyse und Optimierungsbedarf der Übertragungen

Wie im Kapitel 3.3 beschrieben, wird Axis für die Entwicklung von Web Services verwendet und es wird erwähnt, dass Axis bereits in der Version Axis2 vorliegt. Axis2 weist im Vergleich zu Axis viele Vorteile und Verbesserungen auf, die im Kapitel 5.5.1 genannt werden. Deshalb wird eine Umstellung von Axis auf Axis2 empfohlen. Binäre Daten werden für eine Übertragung sowohl von Axis als auch Axis2 mit Base64[3] kodiert.
Die Problemanalyse und die Feststellung von Optimierungsbedarf der Übertragungen wird im Folgenden separat, für die im Kapitel 5.2 genannten Web Services DLRWebservicesFCD und DLRWebservicesRawFCD, vollzogen. Der Datenaustausch wird mit dem TCP-Monitor untersucht. Der Monitor fungiert als Überwachungsprogramm. Dadurch können bei Übertragungen[4] die Protokolle erhalten, gespeichert und analysiert werden. Für die Testdurchführungen wird sowohl ein Client als auch der Server auf ein und demselben Personal Computer betrieben. Zusätzlich kann durch ein Testprogramm die Kommunikation und der Datenaustausch wie in der Praxis betrieben werden. Die entwickelten Programme werden, zusammen mit den im Folgenden erwähnten Dateien und Protokollen, in der Beilage zur Verfügung gestellt. Die Beschreibung für eine Nutzung erfolgt im Anhang E.

[3]Kodierung der binär Daten zu einer Zeichenfolge mit autonomen ASCII-Zeichen
[4]vom Client zum Server oder vom Server zum Client

Analyse DLRWebservicesRawFCD

Für die Testzwecke wird eine Funktion des DLRWebservicesRawFCD gewählt, mit der eine Ansammlung von FCD-Daten angefordert werden kann. Der Server ordert dies einem Client an. Infolgedessen werden beim Client die Bytes einer Datei ausgelesen und dem Server geschickt. Anhand der Protokolle wird festgestellt, dass zu viele Zeichen für verhältnismäßig wenige Informationen verwendet werden. Jede Information wird bei diesem Web Service durch XML dargestellt und mit Tags beschrieben. Diese Informationen werden in einer Textdatei abgespeichert. Der Inhalt der Datei wird für eine Übertragung als Byte-Stream ausgelesen, sodass nur die Bytes und nicht die Zeichen übertragen werden. Diese Handhabung umgeht die Probleme und Konflikte, die durch das Einbinden von XML in SOAP, also auch XML, entstehen würden. Die hohe Zeichenmenge resultiert aus der Eigenheit von XML, jede Information mit öffnenden und schließenden Tags darzustellen. Im Anhang A.2 ist ein Protokoll des TCP Monitors abgebildet, das bei einer Übertragung von Nutzer zu Server verwendet wurde. Für die Übersichtlichkeit wurden über 90000 Zeichen entfernt, die zu den Byte-Daten gehören. Auf Grund dieser Ergebnisse entsteht die Motivation ein geeignetes Beschreibungsformat als Alternative zu XML zu wählen, welches die Informationen ausreichend definiert und die Datenmenge vermindert. Eine Reduzierung der Datenmenge und eine Verbesserung der Zusammensetzung ermöglicht eine Minderung der Übertragungskosten und des Aufwands und gewährt eine schnellere und sicherere Übertragung.

Analyse DLRWebservicesFCD

Für die Testzwecke wird die Anwendung gewählt, um Geschwindigkeiten von Kanten[5] zu übertragen. Ungefähr jeden Monat wird einmal ein beträchtlicher Zusammenschluss an Kanten und zugehörigen Geschwindigkeiten vom Server an den Nutzer gesendet. Es handelt sich um sogenannte historische FCD-Daten. Bei diesem Anwendungsszenario müssen sehr große Datenmengen übertragen werden. Die Daten liegen in einer Binär-Datei vor. Für diese Arbeit wird eine Datei vom Unternehmen zur Verfügung gestellt, die über 37 MB groß ist. Das Institut hat bereits einen Lösungsansatz entwickelt, um die Daten komprimiert zu übertragen. Die Binär-Datei wird mit dem Kompressionsprogramm GZIP verkleinert und daraus entsteht eine Datei mit der Endung .gz und einer Größe von ungefähr 2,8 MB. Die Kompression zeigt sich als eine effektive Variante, indem die Binär-Datei um mehr als das 13-fache reduziert werden kann. Aber eine Kompression fordert zugleich eine Dekomprimierung auf Seiten des Clients. Dadurch kann vereinzelt eine Zusammenarbeit mit Partnern erschwert werden[6]. Zudem entsteht bei der Kompression

[5]Bezeichnung für Identifikationsnummern, die Teile einer Straße abbilden

[6]nicht jeder Nutzer ist bereit und im Bilde entsprechende Kompressionsprogramme zu verwenden

Aufwand[7]. Dennoch ist es empfehlenswert bei dieser Anwendung auf Technik der Kompression zurückzugreifen, da nur selten eine Übertragung stattfindet (ungefähr monateweise) und der Aufwand entsteht. Es wird aber eine enorme Datenreduzierung erreicht. Nachdem mit dem TCP-Monitor ein Protokoll erhalten wurde, kann es ausgewertet werden. Es wird festgestellt, dass zwei Byte-Streams gesendet werden, obwohl nur der Inhalt einer Datei gesendet wird. Für weitere Ausarbeitungen und Entwürfe gilt es zu untersuchen, ob ein Stream ausreichend ist.

5.3. Ausarbeitung und Entwürfe für Verbesserungen der Übertragung

Für den Web Service DLRWebservicesRawFCD gilt es, ein geeignetes Datenübertragunsformat zu wählen, das alternativ zu XML verwendet werden kann. Dabei muss beachtet werden, dass ein gewähltes Format die Anforderungen einer technischen Neutralität für eine SOA erfüllt, damit dynamisches Binding weiterhin unterstützt wird[8]. Zudem soll das Format die Informationen kompakt beschreiben. Das Datenübertragungsfomat JSON (JavaScript Objekt Notation) wird als Technologie identifiziert, die diesen Anforderungen gerecht wird. Im Kapitel 5.4 wird JSON beschrieben und die Vorteile gegenüber XML, im Bezug auf dieses Projekt, dargelegt.
Bei dem Web Service DLRWebservicesFCD werden für die komprimierte Übertragung von historischen FCD-Daten zwei Byte-Streams im Protokoll erkannt. Nach genaueren Untersuchungen wird festgestellt, dass für die Übertragung die Bytes aus der komprimierten Datei ausgelesen werden. Für eine Übertragung werden diese Bytes dekomprimiert als String gesetzt zusätzlich zu den eigentlichen Bytes, also eine Redundanz der Daten. Der dekomprimierte String ist überflüssig und die Datenmenge kann halbiert werden.

5.4. Das Datenübertragungsformat JSON als Alternative zu XML

JSON (JavaScript Objekt Notation) ist ein schlankes Datenaustauschformat, dessen Sprachumfang von JavaScript entnommen wurde[9]. JSON ist eine Notation, deren Datenstruktur Objekte, Arrays, Zeichenketten, Zahlen, boolesche Werte und null enthalten kann. Für fast alle gängigen Programmiersprachen wurden Schnittstellen für die Notation entwickelt [JSON]. Objekte können durch JSON mit sehr wenig

[7]Komprimierung und Dekomprimierung benötigt Zeit und Prozessor-Ressourcen
[8]siehe Kapitel 6.3
[9]deshalb ist im Namen von JSON JavaScript enthalten

Zeichen ausreichend beschrieben werden. Wie die verschiedenen Datenstrukturen von JSON dargestellt werden, wird im Groben, anhand des Beispiels von JSON in Abbildung 5.2, erklärt. Eine umfangreiche Dokumentation kann auf der Webpräsenz von JSON entnommen werden[10]. Person ist in diesem Beispiel ein Objekt, das durch geschweifte Klammern ausgedrückt wird[11]. Der Name eines Objektes[12], ist ein String dem durch Doppelpunkt ein Wert zugeordnet werden kann. Person hat als Wert einen Array an Objekten, die durch Komma getrennt aufgelistet werden können. Ein Array wird durch eckige Klammern gesetzt[13]. Die Objekte des Arrays haben als Wert einen String.
Es entsteht die Motivation für dieses Projekt als Datenübertragungsformat JSON alternativ zu XML zu verwenden, auf Grund der kompakten Schreibweise von JSON und dem großen Angebotsfolio an Schnittstellen. In der Abbildung 5.2 wird JSON im Vergleich zu XML gezeigt. Bei diesem Beispiel benötigt JSON 111 Bytes und XML 133 Bytes für die Beschreibung, sodass der Vorzug von JSON deutlich wird. Hierbei handelt es sich nur um ein kleines Anwendungsbeispiel und es ist anzunehmen, dass bei hohen Datenaufkommen der Nutzen von JSON signifikant erkennbar wird. Im folgenden Kapitel 5.4.1 wird mit prototypischen Daten praxisnah eine Umstellung von XML auf JSON für den Web Service DLRWebservicesRawFCD durchgeführt und bewertet.

JSON	XML

{
“Person“:[{
“Name“:“Müller“,
“Vorname“:“Max“,
“Alter“:“22“,
“Bemerkung“:“Studiert an der TFH“
}
]
}
	<Person>
	<Name>Müller</Name>
	<Vorname>Max</Vorname>
	<Alter>22</Alter>
	<Bemerkung>
	Studiert an der TFH
	</Bemerkung>
	</Person>

Abbildung 5.2.: JSON im Vergleich zu XML

5.4.1. Durchführung und Bewertung einer Umstellung von XML zu JSON

Eine Konvertierung von XML-Datensätzen zu JSON kann zum Beispiel mit Java sehr einfach realisiert werden. Dafür wird von JSON eine Java-Klasse zur Verfü-

[10] siehe [JSON]
[11] für Anfang des Objektes eine geöffnete ({) und für das Ende eine schließende(})
[12] in diesem Beispiel “Person”
[13] für den Anfang des Arrays geöffnete ([) und für das Ende schließende ([)

gung gestellt, die Funktionen bereitstellt um die Konvertierung durchzuführen[14]. Bei dem Web Service DLRWebservicesRawFCD werden die FCD-Roh-Daten durch XML beschrieben und in einer Datei zwischengespeichert. Für weitere Auswertungen oder für eine Übertragung wird dieser Dateiinhalt verwendet. Es wird sowohl der benötigte Speicherplatz für eine Zwischenspeicherung als auch die Übertragungsvolumen deutlich reduziert, wenn die Beschreibung der FCD-Roh-Daten in einem kompakteren Format als XML erfolgt.
Bei der Testdurchführung werden die FCD-Roh-Daten, die in einer Datei zwischengespeichert sind, von XML zu JSON konvertiert und in einer separaten Datei abgespeichert. Anhand der Dateigrößen oder anhand der verwendeten Zeichen für die Beschreibung der Daten, sind die Technologien zu vergleichen. Es wird eine GUI geschrieben, in welcher der Inhalt einer Datei mit FCD-Roh-Daten als XML-Zeichenkette eingelesen und danach zu einer JSON-Zeichenkette konvertiert wird. Diese Zeichenketten können in ihrer Größe verglichen werden. Zudem werden in der Klasse Performance Vorteile schon durch entsprechende Systemausgaben aufgezeigt und in einer Protokoll-Textdatei gespeichert. Das Projekt wird zusammen mit dem Protokoll und den Datendateien als Beilage bereitgestellt. Im Anhang E.3 werden die Anforderungen und Beschreibungen für eine Nutzung dokumentiert. Als Ergbnis wird für diese Anwendung eine Verbesserung, beziehungsweise Verminderung der Zeichen durch JSON um ungefähr 31 Prozent festgestellt. Des Weiteren wird ermittelt, dass die Konvertierung von XML zu JSON schnell erfolgt. Trotzdem wäre es aufwendig eine Konvertierung bei jeder Datenübertragung und -speicherung durchzuführen. Daher empfiehlt es sich Daten einheitlich mit JSON zu beschreiben und JSON als Datenübertragungsformat zu wählen, sodass eine Konvertierung überflüssig wird.

5.5. Entwicklung der Web Services mit Axis2 statt Axis

Die Erstellung eines Web Services mit Axis2 kann sehr einfach und fast automatisch durchgeführt werden, eine detaillierte Beschreibung erfolgt im Kapitel 6.1. Diese Dokumentation dient als Entwurf und Optimierung für weitere Web Services Entwicklungen der Plattform. Eine Verbesserung ist gegeben, indem die einzelnen Prozessabläufe für die Kreation eines neuen Web Services einfach verstanden und sehr schnell durchgeführt werden können. Um eine effektive Umstellung und fähige Integration in die TDP zu gewähren müssen folgende Aspekte beachtet werden:

- Bevor eine Umstellung durchgeführt wird, müssen außer dem Interface und dem Datenmodell, alle Klassen entfernt werden. Aus dem Interface kann mit

[14]die Klasse XML.java, siehe http://www.json.org/java/

Axis2 und Ant eine WSDL-Datei und aus dieser die Klassen erstellt werden. Dies kann mit dem Programm *WSDL2JAVA* erfolgen. Eine detaillierte Beschreibung dieser Code-Generierung wird im Kapitel 6.1.1 genannt.

- Es müssen sowohl auf Seiten des Servers, als auch auf Seiten des Clients Anpassungen für Axis2 vollzogen werden. Die Web Service Funktionen, die eine Businesslogik darstellen, werden mit Axis2 anders repräsentiert. Mit einem Data Binding werden die Daten beschrieben und ein Zugriff ermöglicht. Weitere Informationen dazu können den folgenden Kapiteln entnommen werden.

5.5.1. Vorteile durch Axis2

Im Vergleich zu Axis stehen bei Axis2 viele neue Features bereit. So wird beispielsweise "Hot Deployment" unterstützt. Zur Laufzeit erkennt der Webserver eine Veränderung von registrierten Web Services und führt darauffolgend automatisch eine Aktualisierung durch. Im Vergleich dazu ist bei Axis nach einer Anpassung eines Services ein Neustart des Webservers erforderlich, damit Änderungen übernommen werden. Die Funktionen und Daten werden in Axis2 durch Data Binding beschrieben und serialisiert. Axis2 protegiert AXIOM um XML Elemente anpassen oder hinzufügen zu können [Vgl.: Apache 2009c, Data Binding Support]. AXIOM steht für Axis Object Model und basiert auf die StAX API. StAX (Streaming API for XML) ist eine Technologie, die dafür verwendet wird XML Daten von einer Anwendung zu erzeugen und auch XML für eine Anwendung einzulesen und zu parsen. Entgegen basiert Axis auf SAX (Simple API for XML), das als Schnittstelle für XML Bearbeitung verwendet wird. Gegenüber StAX beschränkt sich SAX nur auf das Einlesen von XML Daten.
Des Weiteren werden durch Axis2 asynchrone Anforderungen und Übertragungen eines Clients möglich. Ein Nutzer kann während er auf die Antwort und Daten des Servers wartet, weitere Transfers inszinieren. Dadurch können bedeutende Performancevorteile erlangt werden. Gerade bei datenintensiven Prozessen ist es wichtig Parallelität bei der Kommunikation zwischen Client und Server zu fördern. Durch eine parallele Verteilung der Prozesse lassen sich zeitaufwendige Durchführungen schneller abarbeiten. Mit Axis2 werden viele weitere bedeutende Web Service Technologien unterstützt, unter Anderem auch JSON. Es wird zum Beispiel ein *JSONOMBuilder*[15] und ein *JSONMessageFormater*[16] bereitgestellt. Aus diesem Grund werden die Ausarbeitungen und Ergebnisse zu JSON in Kapitel 5.4.1 durch Axis2 in der TDP begünstigt.

[15] mit dieser Klasse können auf Basis des Datenaustauschformats JSON die Object Models (OM) gebildet werden

[16] mit dieser Klasse kann der Informationsfluss mit JSON als Alternative zu XML formartiert werden.

6. Systemkonzept für die Prozess- und Dienstoptimierung der SOA Plattform

6.1. Einfache und automatische Entwicklung eines neuen Web Service mit Axis2

Folgend wird ein Konzept beschrieben, um vereinfacht einen neuen Web Service zu entwickeln und in eine SOA einzubinden. Durch diese Beschreibungen, von der Defintion der angebotenen Funktionen bis hin zum Deployment und Veröffentlichen des Web Services, wird eine Entwicklung sehr erleichtert. Die Prozessabläufe für eine Generierung eines neuen Dienstes werden deutlich dargelegt und vereinfachen somit den Vorgang. Die Dokumentation erfolgt beispielhaft anhand des Web Services Smarttruck, der in Kapitel 5.2 für die Optimierung der Übertragung schon erklärt und verwendet wurde. Das gesamte Projekt und alle verwendeten und beschriebenen Skripte sind auf der Beilage enthalten. Im Anhang E.2.1 erfolgen Beschreibungen und Vorraussetzungen für eine Nutzung.
Das Forschungsprojekt Smarttruck verfügt über mehrere POJO (Plain Old Java Object). Ein POJO ist ein normales Java Objekt. Bei Smarttruck sind die Objekte ein POJO, denen die FCD Daten gesetzt werden[1]. Zudem sind die Datenobjekte ein POJO, die bei der Angabe eines Status oder bei den Anforderungen und Anfragen für die Kommunikation verwendet werden[2]. Diese Objekte müssen übertragen werden können.

6.1.1. Automatische Codegenerierung mit Apache Ant

Durch ein Ant Skript kann automatisch von einem Java-Interface[3] eine WSDL-Datei erzeugt werden. Mit einem Ant Skript ist die *build.xml* Datei gemeint, in der wichtige Einstellungen und Tasks definiert werden und die von Ant ausgeführt wird.

[1] zum Beispiel *FCDTrafficDataObject*
[2] zum Beispiel *StatusData* oder *RequestData*
[3] ist eine Schnittstelle mit Funktionen, die aber nicht ausprogrammiert sind

Dafür wird im Skript ein Task[4] angelegt. Der Task verwendet das Programm *JAVA2WSDL*[5], welches der Bibliothek von Axis2 mitgeliefert wird. In der Abbildung 6.1 werden die Argumente gezeigt, die zusätzlich zur Ausführung des Programms gesetzt und mit Werten angegeben werden können. Das Interface beinhaltet die

Argument	Bedeutung
-o	Output Verzeichnis der WSDL
-of	Dateiname der WSDL
-l	Endpunkt unter dem der Service erreichbar sein wird
-cn	Name der Klasse / Interface die den Service repräsentiert
-cp	Classpath. Hier unsere kompilierten Sourcen (Interface + POJO)

Abbildung 6.1.: Mögliche Argumente bei dem Programm JAVA2WSDL
Quelle: [Wilkening Juli 2008]

Funktionen des Web Services, die nun in der WSDL-Datei definiert sind. Doch es fehlen noch wichtige Klassen die für eine Kommunikation zwischen Nutzer und Server benötigt werden. Für den Server wird eine Skeleton Klasse benötigt. Es ist ein Gerüst, das die Logik der Anwendung[6] beinhalten muss. Weiterhin ist eine *services.xml* erforderlich, die den Service und die verwendeten Klassen beschreibt. In der Skeleton Klasse wird die Logik manuell eingetragen. Zudem muss von dem Service eine AAR-Datei (Axis Archive) erstellt werden, die einer JAR sehr ähnelt. Im Kapitel 6.1.2 ist beschrieben, wie der Service bei dem Webserver von Apache Tomcat durch die AAR-Datei hinzugefügt und registriert wird. XML wird dafür verwendet, die Business Daten zu repräsentieren und somit den Web Service zu unterstützen. Auf Grund dessen wird eine Technik benötigt, die eine Umwandlung von XML-Beschreibungen zu Datenstrukturen der Anwendung ermöglicht. Im Zusammenhang mit Axis2 handelt es sich bei einer solchen Technik um Data Binding. Der Client benötigt eine Klasse, die das Data Binding abbildet und somit Übertragungen und die gesendeten Datenstrukturen interpretieren kann. Bei Axis2 wird diese Klasse als Stub bezeichnet, dem unter Anderem die URL des Web Services bekannt sein muss. Ein Programmbeispiel für einen Client kann der Beilage entnommen werden. Für eine Nutzung sollte die Beschreibung im Anhang E.2 beachtet werden. Anhand der WSDL-Datei wird mit dem Ant Skript, durch einen weiteren Task, sowohl der Server als auch der Client Code automatisch generiert. Hierfür wird das Programm *WSDL2JAVA*[7] von Axis2 verwendet. In der Abbildung 6.2 werden die

[4] wird in einem *target* im Skript erstellt.
[5] die Java-Klasse *org.apache.ws.java2wsdl.Java2WSDL*
[6] Mit der Logik der Anwendung ist die Geschäftslogik gemeint
[7] die Java-Klasse *org.apache.axis2.wsdl.WSDL2Java*

Argumente gezeigt, die zusätzlich zur Ausführung des Programms gesetzt und mit Werten angegeben werden können.

Argument	Bedeutung
-uri	Ort der WSDL Datei
-ss	Generierung von serverseitigem Code
-g	Generierung aller Klassen
-sd	Generierung des Service Deskriptor services.xml
-o	Verzeichnis der generierten Java Klassen
-p	Package der Klassen (optional)

Abbildung 6.2.: Mögliche Argumente bei dem Programm WSDL2JAVA
Quelle: [Wilkening Juli 2008]

6.1.2. Einfache Veröffentlichung eines Services

Apache Tomcat fungiert als Container für die Serivces. Entwickler fügen neue Services, die Geschäftsprozesse repräsentieren, dem Framework hinzu. Dabei werden die Dienste von Tomcat als Servlets interpretiert. Tomcat fungiert in Folge dessen bei diesem Projekt als Register für Services der SOA. Nutzer können veröffentlichte Services über eine von Tomcat angebotene Webpräsenz erhalten. Zudem wird die WSDL, die Beschreibung der Web Services, zur Verfügung gestellt. Damit Axis2-Anwendungen bei Tomcat registriert werden können, muss Axis2, als eine Webanwendung, Tomcat hinzugefügt werden. Die *axis2.war* (Web Archive) Datei wird im Tomcat-Verzeichnis im Ordner *webapps* eingefügt. Das Web Archive enthält die Webanwendung von Axis2. Daraufhin wird Axis2 beim nächsten Start von Tomcat installiert und die Webpräsenz von Axis2 kann aufgerufen werden. Neue Services lassen sich über die Webpräsenz und deren angebotenen Web Admin Moduls hinzufügen, welche im Anhang D.1 gezeigt wird. Nachdem ein Service hinzugefügt ist, wird dieser als Dienst in der Webpräsenz geführt. Wie im Anhang D.2 verdeutlicht wird, kann der Service daraufhin aufgerufen und mit seinen Funktionen angezeigt werden.

6.2. Unterstützung der Verwendung heterogener Programmiersprachen

In der heutigen Zeit schließen in Unternehmen große komplexe Systeme die Heterogenität ein. Demzufolge liegen den Komponenten und Anwendungen unterschiedliche Technologien zugrunde. Es spielt keinesfalls der Gedanke von einer weltweit

einheitlichen Programmiersprache eine Rolle, denn es gibt fast immer unterschiedliche Technologien, die eine Aufgabenstellung besser und effektiver lösen. Somit ist Heterogenität nicht als Nachteil anzusehen, soweit sie denn von dem System auch unterstützt wird. Die Herausforderung der Heterogenität und somit auch die Verwendung verschiedener Programmiersprachen lässt sich mit einer Serviceorientierten Architektur lösen. Dabei muss ein einheitliches Kommunikationsprotokoll verwendet werden. Die technische Unabhängigkeit kann in einer SOA dadurch gegeben werden, dass ein Service durch standardisierte Protokolle, wie zum Beispiel auf Basis von XML das Protokoll SOAP, aufgerufen werden kann. Diese technische Neutralität gilt als eine der drei Grundvoraussetzungen für dynamisches Binding von Services. Im Folgenden Kapitel 6.3 wird diese Thematik weiter ausgeführt.

6.3. Voraussetzungen für dynamisches Binding der Services in einer SOA

Als dynamisches Binding bezeichnet man den Zusammenhang, dass Services mit beliebig neuen Services oder Anwendungen zusammengeführt werden können und sie lose gekoppelt vorliegen. Sie sind somit unabhängig und nicht nur einem bestimmten Prozess zugeordnet. Dadurch können beliebig viele neue Funktionen und Anwendungen aus schon bestehenden Services entstehen. Damit dynamisches Binding in einer SOA unterstützt wird, müssen folgende Kriterien laut Literatur [Masak 2007, S. 234] für Services erfüllt sein:

> "***Technische Neutralität*** *- Ein Service muss mittels standardisierter Technologie aufrufbar sein. XML-Protokolle erfüllen diese Voraussetzung.*
> ***Lose Kopplung*** *- Ein Service darf auf Consumerseite kein Wissen über interne Strukturen oder Konventionen verlangen.*
> ***Ortstransparenz*** *- Die Informationen über Definition und Ort von Services müssen in einer Registry vorgehalten und dadurch einer Vielzahl von Consumern verfügbar gemacht werden.*"

7. Ergebnisse und Ausblick

7.1. Beschreibung und Bewertung der Vorgehensweise

Die Vorgehensweise erfolgt für diese Arbeit, von der Systemanforderung bis hin zur Bewertung und Durchführung der Systemintegration und des Nutzens, in Form des V-Modells. Dieses Vorgehensmodell gibt eine standardisierte Reihenfolge der Abläufe und Rollenzuweisung vor. In Abbildung 1.1 wurden die einzelnen Schritte für eine Vorgehensweise aufgezeigt. Es handelt sich um Empfehlungen, sodass es nicht zwingend notwendig ist, jeden Schritt zu beachten. Je nach den Anforderungen eines Projekt können einzelne Schritte übersprungen oder geändert, beziehungsweise angepasst werden. Für dieses Projekt ist kein Unittest erforderlich. Die Systemintegration, Nutzen und Abnahme wird zur Bewertung der Systemintegration und des Nutzens angepasst. Die Schritte werden im Folgendem auf diese Arbeit projeziert und bewertet:

1. **Systemanforderungsanalyse:** Im Kapitel 1.2 werden die Anforderungen genannt. Anhand dieser erfolgt im selbigem Kapitel durch eine Analyse die Zielsetzung.

2. **System- und Softwarearchitektur:** Eine Beschreibung und Untersuchung der Systemarchitektur der TDP erfolgt im Kapitel 3.1. Diese Informationen erleichtern weitere Vorgehen deutlich. Es lassen sich benötigte Verbesserungen und Technologien für Konzepte ableiten.

3. **Systementwurf:** Auf Basis der Beschreibungen und Analyse für Verbesserungspotential der Systemarchitektur können entsprechende Entwürfe konzipiert werden. Es werden Konzepte für die Optimierung der Datenorganisation, Übertragungen und Plattform vorgestellt.

4. **Softwaredesign:** Mit Hilfe des Konzeptes und der Anforderungen kann die Software entworfen werfen. Durch die Vorarbeiten ist klar definiert, welche Funktionen benötigt werden.

5. **Integrationtests:** Die Softwareentwicklung erfolgte immer auf Basis des bestehenden Systems. Diese Herangehensweise erleichert die Integrationstests und erhöht die Wahrscheinlichkeit für eine erfolgreiche Integration. Bei der Modifizierung von bestehenden Anwendungen ist ein Integrationtest meistens überflüssig.

6. **Bewertung Systemintegration / Nutzen:** Für jedes Konzept wird die Systemintegration bewertet, der Nutzen offengelegt und mit Tests belegt. Des Weiteren werden Empfehlungen für Handhabung und Verwendung gegeben.

Diese Vorgehensweise und Vorgaben erleichterten die einzelnen Arbeitsschritte. Vor allem zeigten sich die empfohlenen Reihenfolgen bei der Umsetzung als äußerst vorteilhaft. Jeder Schritt liefert für den Nachfolgenden die benötigten Informationen und Ausarbeitungen. Zusätzlich steigt durch diese Vorgehensweise die Erfolgswahrscheinlichkeit für eine letztendliche Abnahme und Nutzung des Unternehmens.

7.2. Zusammenfassung der Ergebnisse

Für die Optimierung des Datenbankmanagementsystems wird die Partitionierung als eine Technik beschrieben und verwendet, die trotz äußerst hoher Datenaufkommen Übersichtlichkeit und Performancevorteile ermöglicht. Nachdem ein Performance Test Ergebnisse entgegen der Erwartung lieferte, konnten diese Ergebnisse analysiert und Rückschlüsse gewonnen werden. Hiernach erfolgte ein modifizierter Performance Test. Die genauen Nutzenvorteile einer Partitionierung werden anhand dieses Performance Tests dargelegt. Die Effizienz steigt bei Befehlen zum Teil zwischen 200 % und 600 %, indem die Befehle 2- bis 6-fach schneller ausgeführt werden können.
In Bezug auf die Optimierung der Übertragungen kann durch die Vermeidung von Redundanzen 50 % der Datenmenge reduziert werden. Des Weiteren wird durch eine Umstellung von XML auf JSON eine Verminderung der Datenmenge um 31 % belegt. Überträgt eine Anwendung sehr große Datenmengen nur gelegentlich, werden die Daten vor einer Übertragung mit GZIP komprimiert. Bei einer Kompression und Dekompression ensteht Aufwand, der bei eher seltenen Übertragungen in Kauf genommen werden sollte, denn durch GZIP konnte eine Reduzierung der Datengröße um das 13-fache festgestellt werden.
Um neue Web Services sehr schnell und einfach erstellen zu können, wird der Ablauf für eine automatische Erzeugung anhand von Apache Axis2 und Apache Ant dokumentiert.

7.3. Ausblick für zukünftige Optimierungsmöglichkeiten

Um Prozesse hinsichtlich ihrer Abläufe und Aktivitäten zu kontrollieren und zu erfassen, wird für die Zukunft ein verbessertes Geschäftsprozessmanagement empfohlen. Dazu soll Business Process Management (BPM), das in Kapitel 2.3 beschrieben wurde, eingesetzt werden. Um BPM effektiv einsetzen zu können, ist es notwendig die einzelnen Prozessabläufe zu analysieren und zu kennen. Die Sprache BPEL, die in Kapitel 2.2.8 für die Verwendung von Orchestrierung von Web Services und in Kapitel 2.3.5 für die Unterstützung des BPM beschrieben ist, kann zukünftig dafür eingesetzt werden die Prozessabläufe zu dokumentieren und Komposition (Orchestrierung) der Prozesse zu unterstützen. Durch BPEL wird die Steuerung und Kontrolle von Geschäftsprozessen während der Laufzeit durchführbar. Ein Management und eine Dokumentation der Aktivitäten von Prozessen ist erforderlich, um die Komposition der Orchestrierung durchzuführen. Mit der Orchestrierung ergibt sich eine empfohlene Trennung der Prozesslogik und Anwendungsfunktionen, die den eigentlichen Web Service abbilden. Die Prozesslogik beinhaltet den Daten- und Kontrollfluss. Durch diese Trennung wird eine bessere Beobachtung und Einschätzung des Datenflusses möglich. In diesem Zusammenhang soll in Zukunft auch die Technologie BPEL angewendet werden, um Geschäftsprozessmanagement zu unterstützen und Kontrolle und Steuerung in Abhängigkeit von Bedingungen und Zuständen zu ermöglichen. Dafür können in BPEL Variablen definiert werden. Zudem unterstützt BPEL die Orchestrierung, indem Abhängigkeiten von Ergebnissen anderer Prozesse definiert werden können.
Auch für die Serviceorientierte Architektur der TDP zeigen sich für die Zukunft noch weitere mögliche Verbesserungen. Denkbar ist es, dass die Infrastrukur eines ESB, wie in Kapitel 2.1.4 beschrieben, zukünftig in der TDP Anwendung findet. Es wurde ein Projekt ausfindig gemacht, dass für die TDP und deren technologischen Anforderungen sowie für eine Umsetzung mit BPEL genau geeignet wäre - das Open Source[1] Projekt JBossESB. Dieses Projekt bietet in der Infrastruktur zum Beispiel zu einer ESB eine Orchestrierungs Umgebung mit BPEL an und in Business Service Komponenten eine Unterstützung für Web Services und POJO's. Eine ausführliche Dokumentation dieses Projektes ist der Seite des Anbieters zu entnehmen [Vgl.: JBoss 2009].
Die zukünftigen Betrachtungen und Ausarbeitungen machen deutlich, dass noch viel Potential besteht, weitere Optimierungen für die Plattform in Zukunft durchzuführen.

[1] wurde im Zusammenhang mit MySQL bereits im Kapitel 2.4.2 erläutert

7.4. Persönliches Fazit

Die Thematik Umgang und Optimierungsmöglichkeiten datenintensiver Prozesse in einer SOA ist ausgeprägt und weit gefächert. Die Materie ist fortschrittlich, interessant und ihr wird für die nächsten Jahre große Nachfrage, Wachstum und Weiterentwicklung nachgesagt. Im Rahmen einer Bachelorarbeit müssen für ein Projekt spezifische Prioritäten gesetzt werden, da allerhand Potential zur Verfügung steht, aber leider nicht die Zeit alles auf einmal umzusetzen.
Eine Eigenschaft der Informationstechnik (IT) ist es, dass in der Regel entwickelte Software nicht auf Anhieb einwandfrei funktioniert, sondern immer wieder Anpassungen und Verbesserungen durchgeführt werden müssen. Manchmal können schon minimale Einstellungen, Variablen, Parameter und ähnliche, die inadäquat verwendet werden, zu Inkompatibilitäten oder Beeinträchtigungen des Programmes führen. Diese können bei geringfügigen Problemen Ergebnisse verfälschen und bei erheblichen Fehlern ein Programm nicht ausführbar machen. Bei komplexen Projekten kann es vereinzelt mehrere Stunden oder gar Tage dauern, um die Ursache ausfindig zu machen. Die Herausforderung zeigte sich auch bei dieser Arbeit, indem beispielsweise die partitionierten Tabellen zuerst Ergebnisse entgegen den Erwartungen lieferten. Wie bei diesem Beispiel gezeigt wurde, empfiehlt es sich Annahmen für die Fehlerusachen aufzustellen und diese zu prüfen. Eine weitere Eigenheit der IT ist es, dass sie einem ständigen Wandel unterliegt. Mit der Zeit gibt es immer wieder neue und erweiterte Technologien. Für IT-Spezialisten stellt sich als weitere Herausforderung jederzeit auf dem aktuellen Stand sein zu müssen, um bei Unternehmungen die besten Lösungen anwenden, konzipieren und implementieren zu können.
Aber genau das macht die IT erst so einzigartig interessant, wenn man sich dieser Herausforderungen annimmt.

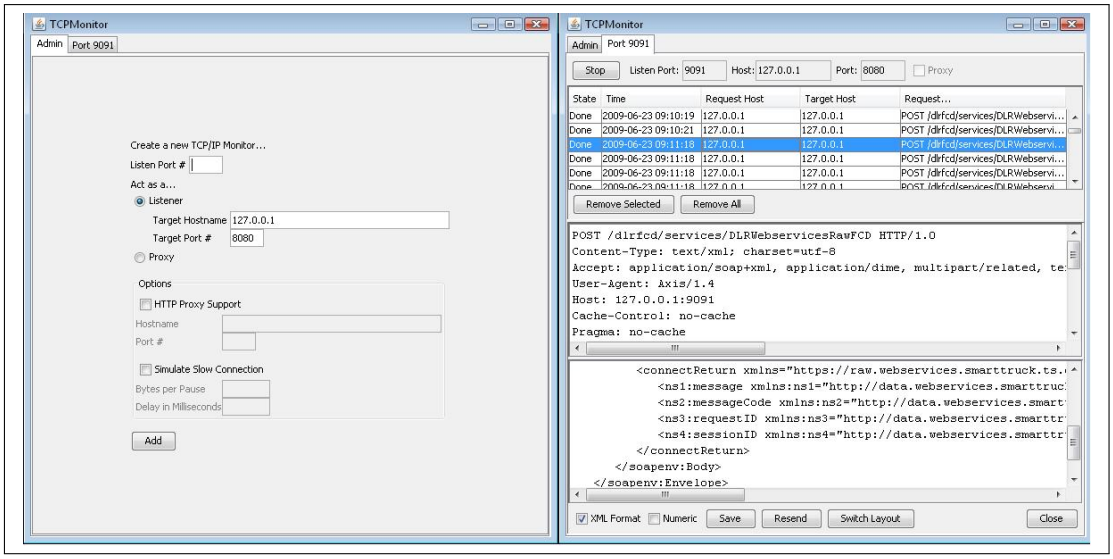
8. Zusammenfassung

In der heutigen Zeit streben Unternehmen nach Systemen, die hohe Flexibilität, Agilität, Interoperabilität und Dynamik bieten. Dafür werden Paradigmen inszeniert und gefördert. In diesem Zusammenhang zeigt sich das Systemparadigma einer SOA als begehrt. Um Interoperabilitätsprobleme zu lösen und unternehmensübergreifend agieren zu können, realisieren viele Unternehmen eine SOA mit Web Services. Dabei stellen datenintensive Prozesse eine bedeutende Herausforderung dar. Es wird der richtige Umgang mit den großen Datenmengen abgefordert. Mit dieser Arbeit wurde die Herausforderung angenommen und speziell für die Traffic-Data-Platform, des Instituts für Verkehrssystemtechnik des DLR e.V., adäquat Lösungen konzipiert und implementiert. Auf Basis der beschriebenen Grundlagen für diese Thematik, sind Ergebnisse für das Datenbankmangementsystem, die Datenübertragungen und Web Services der SOA vorgestellt. In Bezug auf das Datenbankmanagementsystem ist die Partitionierung von Datenbanktabellen erläutert und wird angewendet. Als Vorteil einer Parititonierung ist ausgewiesen, dass trotz äußerst hoher Datenaufkommen die Übersichtlichkeit und Performancevorteile ermöglicht werden. Für die Optimierung von Übertragungsmengen wurde JSON, die kompakte Objekt Notation, als geeignetes Übertragungsformat alternativ zu XML vorgestellt und der Vorteil dieser Notation belegt. Überträgt eine Anwendung sehr große Datenmengen nur gelegentlich, werden die Daten vor einer Übertragung mit GZIP komprimiert. Bei einer Kompression und Dekompression ensteht Aufwand, der bei eher seltenen Übertragungen in Kauf genommen werden sollte, denn durch GZIP konnte eine Reduzierung der Datengröße um das 13-fache festgestellt werden. Des Weiteren ist die vereinfachte und automatische Erzeugung von Web Services mit Apache Axis2 und Apache Ant dargestellt. Für zukünftige Betrachtungen sind Informationen von Geschäftsprozessmanagement und Orchestrierung von Geschäftsprozessen mit BPEL dokumentiert.

Anhang

A. Datenübertragung

A.1. Das Programm TCP Monitor

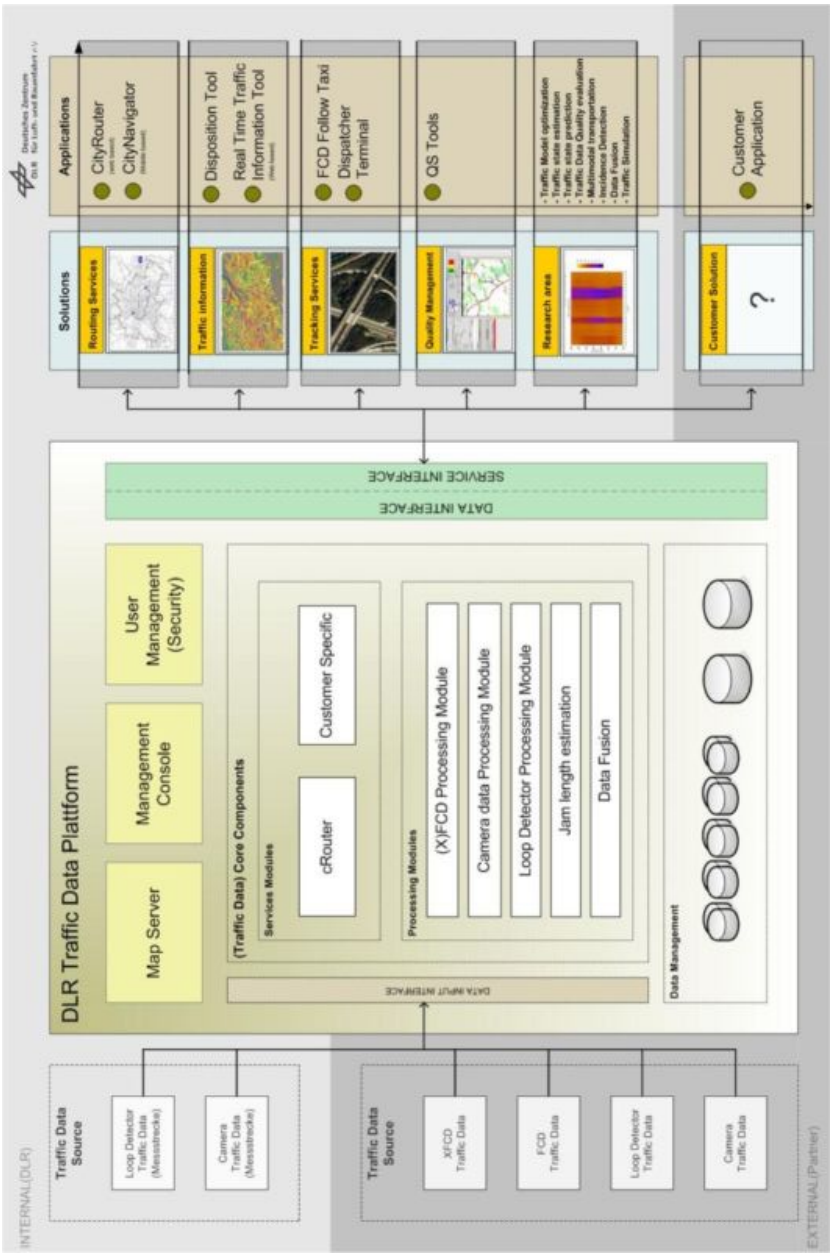


A.2. Ein Protokoll des DLRWebservicesRawFCD

```
==============
Listen Port: 9091
Target Host: 127.0.0.1
Target Port: 8080
==== Request ====
POST /dlrfcd/services/DLRWebservicesRawFCD HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related,
text/*
User-Agent: Axis/1.4
Host: 127.0.0.1:9091
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 94615

<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><soapenv:Body>
<in2 xmlns="https://raw.webservices.smarttruck.ts.dlr.de"><ns1:rawFCD
xmlns:ns1="http://data.webservices.smarttruck.ts.dlr.de">
UkVTVUxUPjxFUlJPUj48SUQ+MDwvSUQ+PC9FUlJPUj48UEFSQU1FVEVSPjxLVU5ERV9OQ
U1FPkRMUjwvS1VOREVfTkFNRT48REVGQVVMVFNQUkFDSEVfQ09ERT5ERTwvREVGQVVMVF
...[above 90000 similar charts]...
+MzI8L1NFS1VOREVOPjxTTOxMWkVJVD4xMDwvUO9MTFpFSVQ+PEFCRkFIUlQ+PFg+MTEu
NTD48LOZBSFJUPjwvR1BTRkFIUlRTVEFUSVNUSUs+PC9QQVJBTUVURVI+PC9SRVNVTFQ+
</ns1:rawFCD><ns2:requestData xmlns:ns2
="http://data.webservices.smarttruck.ts.dlr.de"><ns2:requestID>1
</ns2:requestID><ns2:sessionID>-4260892860702943492</ns2:sessionID>
</ns2:requestData><ns3:timestamp xmlns:ns3
="http://data.webservices.smarttruck.ts.dlr.de">1245741018281
</ns3:timestamp></in2></soapenv:Body></soapenv:Envelope>
```

B. Traffic-Data-Platform

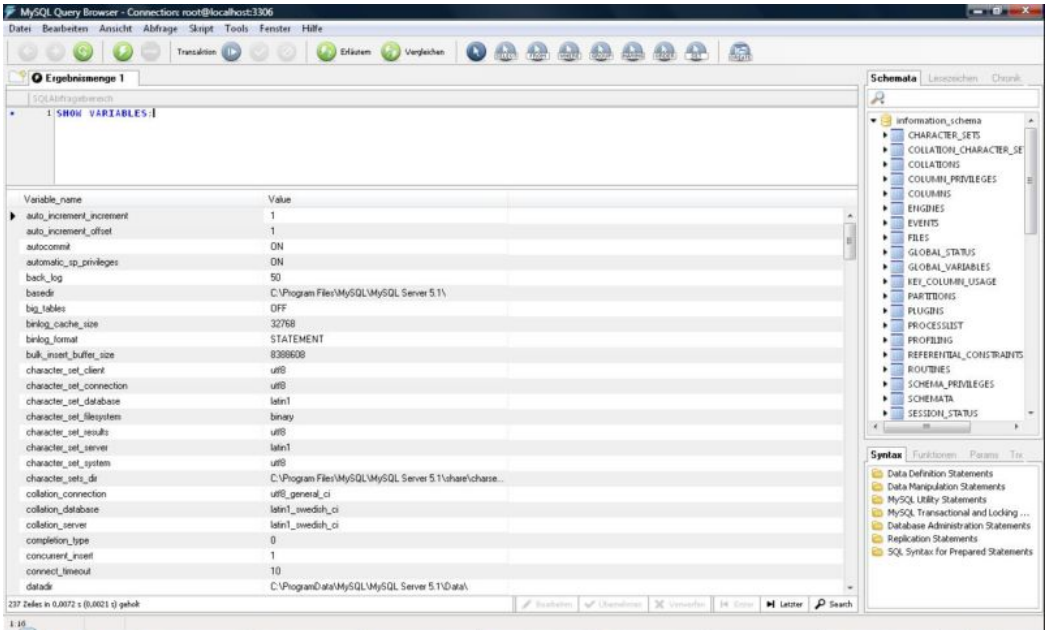


C. Datenorganisation

C.1. Angabe der Testumgebung

Eigenschaft	Wert
Betriebssystem	Microsoft Windows Server 2003
Betriebssystem Version	Standard x64 Edition, Service Pack 2
Prozessor	Intel Xeon CPU E5 20 2,27 GHz
Arbeitsspeicher	23,9 GB RAM
Festplattenspeicher	Mehr als 1 TB frei
MySQL Version	5.1.34

C.2. Screenshot des verwendeten MySQL Query Browser



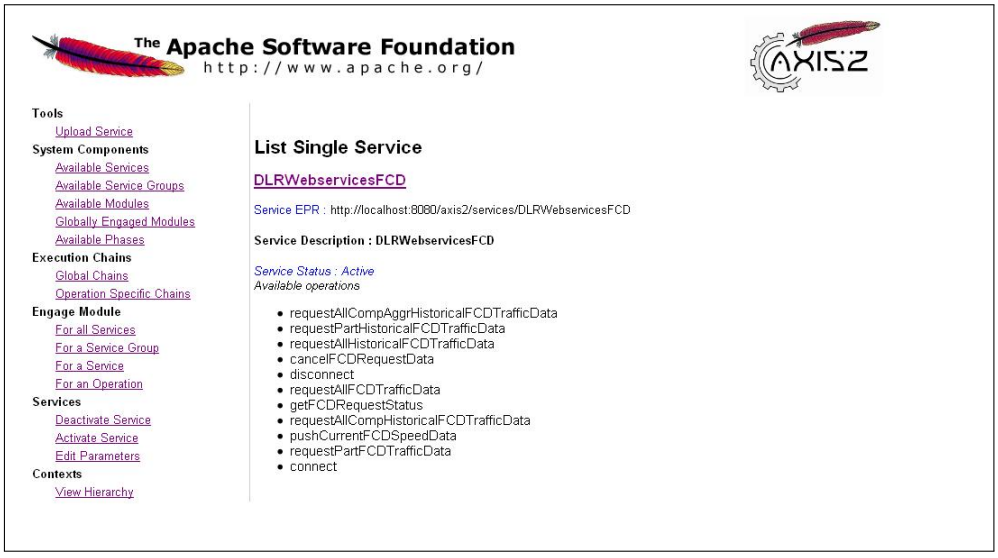
C.3. Performancetest der Tabellen

Query	Non Partition 2008	Non Partition 2009	Non Partition 2008 & 2009	Partition Typ 1	Partition Typ 2	Partition Typ 3	Partition Typ 4
SELECT * FROM x WHERE year(date) BETWEEN 2009 AND 2010 AND month(date) BETWEEN 4 AND 5; ->2860465 Zeilen	* 0 Zeilen. Nur die Zeit des Jahres 2008 1.Durchführung: 0,0006s (19,8199s) 2.Durchführung: 0,0005s (19,5768s) 3.Durchführung: 0,0004s (19,5998s)	1.Durchführung: 22,9948s (0,0016s) 2.Durchführung: 22,9652s (0,0019s) 3.Durchführung: 23,0378s (0,0017s)	1.Durchführung: 29,6001s (19,6845s) 2.Durchführung: 29,6743s (19,6486) 3.Durchführung: 29,3182s (19,6967s)	1.Durchführung: 24,7047s (20,6665s) 2.Durchführung: 24,5448s (20,5975s) 3.Durchführung: 24,5920s (20,5578s)	1.Durchführung: 20,6641s (22,6600s) 2.Durchführung: 22,3882s (22,6604s) 3.Durchführung: 21,4994s (22,6229s)	1.Durchführung: 24,6822s (20,7659s) 2.Durchführung: 24,8425s (20,7016s) 3.Durchführung: 23,9404s (20,6680s)	1.Durchführung: 20,9627s (23,4865s) 2.Durchführung: 21,4173s (23,2415s) 3.Durchführung: 21,0685s (23,1873s)
SELECT * FROM x ORDER BY month(date) LIMIT 1000; ->1000 Zeilen	1.Durchführung: 0,0047s (29,1468s) 2.Durchführung: 0,0059s (32,3171s) 3.Durchführung: 0,0048s (32,5484s)	1.Durchführung: 0,0055s (39,4006s) 2.Durchführung: 0,0060s (41,1838s) 3.Durchführung: 0,0058s (39,6043s)	1.Durchführung: 0,0047s (33,5818s) 2.Durchführung: 0,0046s (33,5130s) 3.Durchführung: 0,0067s (30,3472s)	1.Durchführung: 0,0058s (35,2216s) 2.Durchführung: 0,0054s (36,1589s) 3.Durchführung: 0,0053s (36,2859s)	1.Durchführung: 0,0048s (37,5665s) 2.Durchführung: 0,0054s (36,1589s) 3.Durchführung: 0,0071s (38,0340s)	1.Durchführung: 0,0049s (35,9368) 2.Durchführung: 0,0056s (35,6784s) 3.Durchführung: 0,0052s (36,8164s)	1.Durchführung: 0,0050s (36,2323s) 2.Durchführung: 0,0067s (36,7010s) 3.Durchführung: 0,0068s (37,0361s)
SELECT * FROM x WHERE date BETWEEN 20081212 AND 20090112; ->2086487 Zeilen	* 1600630 Zeilen. Nur die Zeit des Jahres 2008. 1.Durchführung: 12,345 3s (0,0022s) 2.Durchführung: 12,5422s (0,0019s) 3.Durchführung: 12,3622s (0,0019s)	* 405843 Zeilen. Nur die Zeit des Jahres 2009. 1.Durchführung: 25,2166 (0,0020s) 2.Durchführung: 25,1145 (0,0020s) 3.Durchführung: 25,1985 (0,0020s)	1.Durchführung 15,8290 (0,0021s) 2.Durchführung: 15,4449s (0,0020s) 3.Durchführung: 15,4868s (0,0020s)	1.Durchführung: 15,8594s (0,0020s) 2.Durchführung: 15,6806s (0,0022s) 3.Durchführung: 16,1036s (0,0020s)	1.Durchführung: 16,0128s (0,0024s) 2.Durchführung: 15,8141s (0,0021s) 3.Durchführung: 15,7145s (0,0023s)	1.Durchführung 15,7801s (0,0022s) 2.Durchführung 16,2815s (0,0020s) 3.Durchführung 16,1971s (0,0022s)	1.Durchführung: 15,8800s (0,0025s) 2.Durchführung: 15,7603s (0,0021s) 3.Durchführung: 15,7111s (0,0021s)
SELECT (dayofweek(date)+5)%7 'dayofweek', hour(date) 'date', floor(minute(date)/20)*20 'minute', sum(speed) 'speed', count(*) 'hits' FROM x WHERE date BETWEEN 20081001 AND 20090301 GROUP BY (dayofweek(date)+5)%7, hour(date),floor(minute(date)/20)*20; ->504 Zeilen	* 504 Zeilen. Nur die Zeit des Jahres 2008. 1.Durchführung: 0,0026s (45,8924s) 2.Durchführung: 0,0025s (45,9647s) 3.Durchführung: 0,0029s (45,9120s)	* 504 Zeilen. Nur die Zeit des Jahres 2009. 1.Durchführung: 0,0029s (7,8401s) 2.Durchführung: 0,0030s (7,8467s) 3.Durchführung: 0,0032s (7,8353s)	1.Durchführung 0,0034s (46,3685s) 2.Durchführung: 0,0035s (46,3015s) 3.Durchführung: 0,0029s (47,8655s)	1.Durchführung: 0,0026s (60,4683s) 2.Durchführung: 0,0023s (58,9497s) 3.Durchführung: 0,0022s (58,8528s)	1.Durchführung: 0,0026s (58,9027s) 2.Durchführung: 0,0023s (59,0686) 3.Durchführung: 0,0026s (59,3055s)	1.Durchführung 0,0026s (59,1050s) 2.Durchführung 0,0021s (59,1854s) 3.Durchführung 0,0019s (58,9513s)	1.Durchführung: 0,0025s (58,8152s) 2.Durchführung: 0,0025s (59,0508s) 3.Durchführung: 0,0020s (58,8890s)

D. Service-Entwicklung - Tomcat und Axis2

D.1. Einfacher Upload eines neuen Services

D.2. Anzeige eines verfügbaren Services



E. Beilage - Beschreibungen und Vorraussetzungen für eine Nutzung

Die im Folgenden genannten Projekte werden auf der Compact Disc (CD) bereitgestellt, die als Beilage dieser Arbeit angefügt ist. Die Projekte, die im Anhang E.2, E.3 und E.2.1 für eine Nutzung beschrieben sind, wurden in der Entwicklungsumgebung Eclipse erstellt. Es wurde die Version 3.4.2 von Eclipse verwendet. Um die Projekte einfach nutzen zu können, wird empfohlen Eclipse anzuwenden und die Projekte zu importieren.

E.1. Partitionierung von Datenbanktabellen

Diese Entwicklung setzt folgende Technologien voraus: MySQL Datenbankserver (mindestens Version 5.1.6), ein Programm um die MySQL Befehle durchzuführen (zum Beispiel MySQL Query Browser). Im Verzeichnis *Datenorganisation* werden Dateien zur Verfügung gestellt, die einige verwendete Befehle beinhalten, aber die Dateninhalte der Datenbank, die bei dem Performance Test verwendet wurden, können nicht bereitgestellt werden. In der Datei *DatenorganisationAbfragen.sql* werden angewendete Befehle aufgelistet, bespielhaft für eine Datenbanktabelle.

E.2. Smarttruck - Client und Server Anwendungen

Diese Entwicklung setzt folgende Technologien voraus: Apache Axis2, Apache Tomcat als Servlet Container, Apache Ant und Java 1.5 oder größer. In dem Verzeichnis *smarttruck* befinden sich die Projekte für einen Server (*smarttruck/Server*) und für einen Client (*smarttruck/Client*). In den Projekten werden unter Anderem die verwendete Bibliothek von Axis2 und Ant, in dem Verzeichnis */WEB-INF/lib* zur Verfügung gestellt. Der Java Sourcecode befindet sich im Ordner */WEB-INF/src*. Javadoc befindet sich im Verzeichnis */doc*.

E.2.1. Web Service automatisch erstellen mit Apache Axis2 und Ant

Die Datei *smarttruck/Server/build.xml* ist das Skript, das mit Ant ausgeführt werden kann. In Eclipse lässt sich eine Ansicht für Ant öffnen. Hier kann das Ant Skript hinzugefügt werden und spezielle Tasks ausgewählt werden.

E.3. Smarttruck - XML zu JSON

Diese Entwicklung setzt folgende Technologien voraus: Java 1.5 oder größer. Wie im Anhang E genannt, wird empfohlen Eclipse als Entwicklungsumgebung zu verwenden. In dem Verzeichnis *smarttruck/JsonTest* befindet sich das Projekt. Javadoc

befindet sich im Ordner */doc* und der Java Sourcecode Im Verzeichnis */src*. Die Datendateien und das Protkoll befinden sich im Ordner */res* des Projektes.

Abbildungsverzeichnis

Literaturverzeichnis

[Apache 2009a] Apache: Ant. In: *The Apache Ant Project* (2009). – URL http://ant.apache.org/. – Zugriffsdatum: Juli 2009

[Apache 2009b] Apache: Axis2. In: *The Apache Software Foundation - Axis2* (2009). – URL http://ws.apache.org/axis2. – Zugriffsdatum: Juli 2009

[Apache 2009c] Apache (Veranst.): *Migrating from Apache Axis 1.x to Axis2.* 2009. – URL http://ws.apache.org/axis2/1_5/migration.html. – Zugriffsdatum: Juli 2009

[Barros u. a. März 2005] Barros, Alistair ; Dumas, Marlon ; Oaks, Phillipa: A Critical Overview of the Web Services Choreography Description Language / BPtrends. URL http://www.bptrends.com/publicationfiles/03-05%20WP%20WS-CDL%20Barros%20et%20al.pdf, März 2005. – Forschungsbericht. Zugriffsdatum: Juli 2009

[Blogspan Magazine März 2009] Blogspan Magazine, by Web I.: SOA bringt Wettbewerbsvorteile in schwierigen Zeiten. In: *Blogspan Magazine* (März 2009). – URL http://www.blogspan.net/2812-soa-bringt-wettbewerbsvorteile-in-schwierigen-zeiten.html. – Zugriffsdatum: Juli 2009

[Dostal u. a. 2005] Dostal, W. ; Jeckle, M. ; Melzer, I. ; Zengler, B.: *Service-orientierte Architekturen mit Web Services.* 1. Auflage. Elsevier - Spektrum Akademischer Verlag, 2005. – ISBN 3827414571

[Eberhart und Fischer 2003] Eberhart, Andreas ; Fischer, Stefan: *Web Services - Grundlagen und praktische Umsetzung mit J2EE und .NET.* Carl Hanser Verlag München, 2003. – ISBN 3446227741

[Gudgin u. a. April 2007] Gudgin, Martin ; Hadley, Marc ; Mendelsohn, Noah ; Moreau, Jean-Jacques ; Nielsen, Henrik F. ; Karmarkar, Anish ; Lafon, Yves: SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). In: *W3C - Recommendation* (April 2007). – URL http://www.w3.org/TR/soap12-part1/. – Zugriffsdatum: Juli 2009

[Haas 2003] Haas, Hugo: Service-Oriented Architecture. In: *The World Wide Web Consortium (W3C)* (2003). – URL http://www.w3.org/2003/Talks/0521-hh-wsa/slide5-0.html. – Zugriffsdatum: Juli 2009

[Haas und Brown 2004] Haas, Hugo ; Brown, Allen: Web Services Glossary. In: *The World Wide Web Consortium (W3C) Working Group Note* (2004). – URL http://www.w3.org/TR/ws-gloss/. – Zugriffsdatum: Juli 2009

[It-Research u. a. 2009] It-Research ; Team, Wolfgang M. ; TU, Darmstadt: SOA Check 2009 / T-Systems, Cordys, SAP, Informatica. URL http://www.soa-check.eu/, 2009. – Forschungsbericht. Zugriffsdatum: Juli 2009

[iXenso Software-Solutions e.K 2007] iXenso Software-Solutions e.K (Veranst.): *Die Bedeutung eines Workflowsystems*. 2007. – URL http://www.workflowsysteme.de/html/bedeutung.htm. – Zugriffsdatum: Juli 2009

[JBoss 2009] JBoss (Veranst.): *JBossESB - Reliable SOA infrastructure*. 2009. – URL http://jboss.org/jbossesb/. – Zugriffsdatum: Juli 2009

[Josuttis 2008] Josuttis, Nicolai: *SOA in der Praxis - System-Design für verteilte Geschäftsprozesse*. dpunkt.verlag, 2008. – ISBN 978-3898644761

[JSON] JSON (Veranst.): *Introducing JSON*. – URL http://www.json.org/. – Zugriffsdatum: Juli 2009

[Kavantzas u. a. 2004] Kavantzas, Nickolas ; Burdett, David ; Ritzinger, Gregory ; Fletcher, Tony ; Lafon, Yves: Web Services Choreography Description Language Version 1.0. In: *W3C Working Draft* (2004). – URL http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/. – Zugriffsdatum: Juli 2009

[Kipp 2006] Kipp, Alexander: *Ablösung von WS-CDL durch BPEL und WSFL Global Model*, Universität Stuttgart, Diplomarbeit, 2006. – URL http://elib.uni-stuttgart.de/opus/volltexte/2006/2896/pdf/DIP_2379.pdf. – Zugriffsdatum: Juli 2009

[Kupsch 2006] Kupsch, Florian: *Framework zur dezentralen Integration systemübergreifender Geschäftsprozesse*. EUL Verlag, 2006. – ISBN 978-3899365184

[Manhart März 2007] Manhart, Klaus: Web Services implementieren mit WSDL. In: *Tecchannel* (März 2007). – URL http://www.tecchannel.de/webtechnik/soa/464653/web_services_implementieren_mit_wsdl/. – Zugriffsdatum: Juli 2009

[Masak 2007] Masak, Dieter: *SOA? - Serviceorientierung in Business und Software*. Springer Verlag, Berlin, 2007. – ISBN 978-3540718727

[Microsoft 2009] Microsoft (Veranst.): *Business Activity Monitoring*. 2009. – URL http://www.microsoft.com/germany/bpm/loesungen/bam.mspx. – Zugriffsdatum: Juli 2009

[Microsystems April 2009] Microsystems, Sun: Oracle to Buy Sun. In: *Sun Microsystems Press Room* (April 2009). – URL http://www.sun.com/aboutsun/pr/2009-04/sunflash.20090420.1.xml. – Zugriffsdatum: Juli 2009

[Mohamed 2007] Mohamed, Jana: *SOA im OP - Workflow-Automatisierung auf Basis eines Integrationservers*, TFH Wildau, Diplomarbeit, Dezember 2007

[MySQL 2008] MySQL: Sun to Acquire MySQL. In: *MySQL News & Events* (2008). – URL http://www.mysql.com/news-and-events/sun-to-acquire-mysql.html. – Zugriffsdatum: Juli 2009

[MySQL August 2009a] MySQL: MySQL 5.1 Reference Manual. In: *Documentation* (August 2009). – URL http://dev.mysql.com/doc/refman/5.1/en/. – Zugriffsdatum: August 2009. – revision: 16028

[MySQL August 2009b] MySQL: MySQL 5.1 Referenzhandbuch. In: *Dokumentation* (August 2009). – URL http://dev.mysql.com/doc/refman/5.1/de/. – Zugriffsdatum: August 2009. – Revision: 553

[MySQL August 2009c] MySQL: MySQL 6.0 Reference Manual. In: *Documentation* (August 2009). – URL http://dev.mysql.com/doc/refman/6.0/en/. – Zugriffsdatum: August 2009. – revision: 16028

[Natis 16. April 2003] Natis, Yefim V.: Service-Oriented Architecture Scenario / Gartner. URL http://www.gartner.com/DisplayDocument?doc_cd=114358, 16. April 2003. – Forschungsbericht. Zugriffsdatum: Juli 2009

[Nissen u. a. 29. Mai 2008] Nissen, Volker ; Petsch, Mathias ; Schorcht, Hagen: *Service-orientierte Architekturen - Chancen und Herausforderungen bei der Flexibilisierung und Integration von Unternehmensprozessen*. Gabler Verlag, 29. Mai 2008. – ISBN 978-3835008151

[Oasis 2007] Oasis: OASIS Web Services Business Process Execution Language (WSBPEL). In: *Oasis-Committee* (2007). – URL http://www.oasis-open.org/committees/wsbpel/. – Zugriffsdatum: Juli 2009

[Oracle Dezember 1997] Oracle (Veranst.): *Oracle 8 Release Documentation*. Dezember 1997. – URL http://www.oracle.com/technology/documentation/oracle8.html. – Zugriffsdatum: Juli 2009

[PMH 2008] PMH (Veranst.): *Projektphasen und Meilensteine*. 2008. – URL http://www.projektmanagementhandbuch.de/cms/projektplanung/projektphasen-und-meilensteine/. – Zugriffsdatum: Juli 2009

[Reitbauer und Novakovic 2009] Reitbauer, Alois ; Novakovic, Mirko: Performance First. In: *Java Magazin, Ausgabe März* (2009), S. 72–77

[Ryman u.a. Juni 2007] Ryman, Arthur ; Chinnic, Roberto ; Moreau, Jean-Jacques ; Weerawarana, Sanjiva: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. In: *W3C - Recommendation* (Juni 2007). – URL http://www.w3.org/TR/wsdl20/. – Zugriffsdatum: Juli 2009

[SoaWorld Juni 2007] SoaWorld: Jeffrey Walker, Founder and CTO of TenFold Corporation, to Speak at SOA World 2007. In: *Soa News Desk* (Juni 2007). – URL http://www.soaworld2007.com/node/392049. – Zugriffsdatum: Juli 2009

[Sosnoski Juli 2007] Sosnoski, Dennis: Java Web Services: Axis2 Data Binding / IBM. URL http://www.ibm.com/developerworks/webservices/library/ws-java3, Juli 2007. – Forschungsbericht. Zugriffsdatum: Juli 2009

[Sun 2002] Sun: Jar File Overview. In: *Java Archive (JAR) Features* (2002). – URL http://java.sun.com/j2se/1.4.2/docs/guide/jar/. – Zugriffsdatum: Juli 2009

[Thiele und Habich 2007] Thiele, Maik ; Habich, Dirk: *Orchestrierung datenintensiver Prozesse - Einsatz von BPEL in der Genexpressionsanalyse*. VDM Verlag Dr. Müller, 2007. – ISBN 978-3836402392

[Weigend September 2006] Weigend, Wolfgang: SOA und BPM - Ein perfektes Paar? In: *AP Verlag* (September 2006). – URL http://www.ap-verlag.de/Online-Artikel-Start.htm. – Zugriffsdatum: Juli 2009

[Wilkening Juli 2008] Wilkening, Oliver: SOA II: Automatisierte Erstellung eines Web Service mit Apache Axis2 und Ant / Naxos Software. URL http://www.naxos-software.de/blog/index.php?/archives/56-SOA-II-Automatisierte-Erstellung-eines-Web-Service-mit-Apache-Axis2-und-Ant.html, Juli 2008. – Forschungsbericht. Zugriffsdatum: Juli 2009

[WinterGreen April 2008] WinterGreen, Research: Services Oriented Architecture (SOA) Infrastructure Market Shares Strategies, and Forecasts, 2008 to 2014 / WinterGreen Research. URL http://www.wintergreenresearch.com/reports/SOA%20Engines.html, April 2008. – Forschungsbericht. Zugriffsdatum: Juli 2009

[Woolf 2006] Woolf, Bobby: WebSphere SOA and J2EE in Practice. In: *IBM Developer Blog* (2006). – URL http://www.ibm.com/developerworks/blogs/page/woolf?entry=websphere_enterprise_service_bus. – Zugriffsdatum: Juli 2009

[Ziegler 04. März 2008] Ziegler, Stephan: BITKOM startet Online-Plattform zu SOA / BITKOM. URL http://www.bitkom.org/de/themen/54942_50874.aspx, 04. März 2008. – Forschungsbericht. Zugriffsdatum: Juli 2009